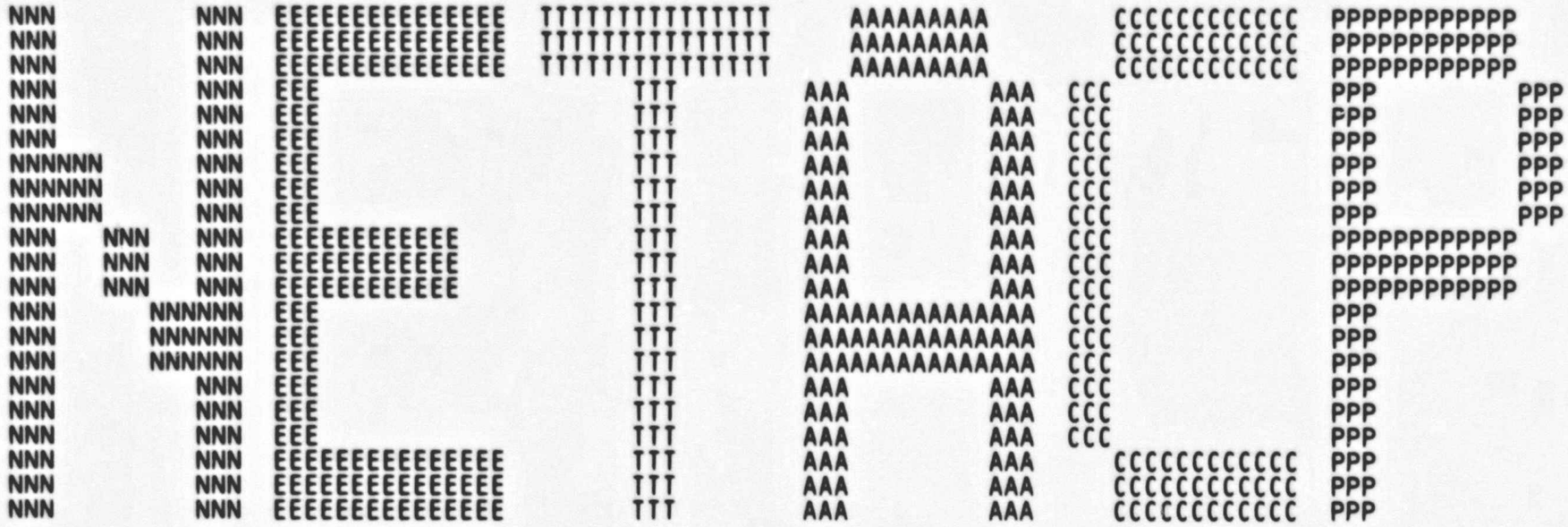
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP

FILEID**NETDRVSES

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    VV      VV    SSSSSSSS  EEEEEEEEEE    SSSSSSSS
NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    VV      VV    SSSSSSSS  EEEEEEEEEE    SSSSSSSS
NN      NN  EE              TT      DD      DD  RR      RR  VV      VV  SS          EE            SS
NN      NN  EE              TT      DD      DD  RR      RR  VV      VV  SS          EE            SS
NNNN    NN  EE              TT      DD      DD  RR      RR  VV      VV  SS          EE            SS
NNNN    NN  EE              TT      DD      DD  RR      RR  VV      VV  SS          EE            SS
NN  NN  NN  EEEEEEEE        TT      DD      DD  RRRRRRRR    VV      VV    SSSSSS    EEEEEEEE        SSSSSS
NN  NN  NN  EEEEEEEE        TT      DD      DD  RRRRRRRR    VV      VV    SSSSSS    EEEEEEEE        SSSSSS
NN    NNNN  EE              TT      DD      DD  RR  RR      VV      VV          SS  EE                    SS
NN    NNNN  EE              TT      DD      DD  RR  RR      VV      VV          SS  EE                    SS
NN      NN  EE              TT      DD      DD  RR    RR      VV  VV            SS  EE                    SS      ....
NN      NN  EE              TT      DD      DD  RR    RR      VV  VV            SS  EE                    SS      ....
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    RR      RR      VV      SSSSSSSS    EEEEEEEEEE  SSSSSSSS      ....
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    RR      RR      VV      SSSSSSSS    EEEEEEEEEE  SSSSSSSS      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  NETDRVSES - DECnet Session Control Module for NETDRIVER
0000     2          .IDENT  'V04-000'
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26
0000    27 ;++
0000    28 ; FACILITY:     DECNET
0000    29 ;
0000    30 ; ABSTRACT:     This module is part of NETDRIVER and is the interface between
0000    31 ;               the user and the NSP layer.
0000    32 ;
0000    33 ; ENVIRONMENT:  KERNEL mode, normal driver environment.
0000    34 ;
0000    35 ;--
0000    36
```

```
0000     38 .SBTTL  HISTORY
0000     39
0000     40 ;
0000     41 ; AUTHOR:       Alan D. Eldridge,  CREATION DATE: 11-JUN-79
0000     42 ;
0000     43 ; MODIFIED BY:
0000     44 ;
0000     45 ;       V03-022 LMP0308         L. Mark Pilant,         31-Aug-1984  16:15
0000     46 ;               Change default state of the ACL queue in the ORB.
0000     47 ;
0000     48 ;       V03-021 ADE1042         A. Eldridge             23-Aug-1984
0000     49 ;               Don't create an XWB if the RCB$W_MCOUNT is zero.  This condition
0000     50 ;               indicates that NETACP is going away and the test is needed to
0000     51 ;               avoid a race condition that can crash the system.
0000     52 ;
0000     53 ;       V03-020 ADE1041         A. Eldridge             25-Jun-1984
0000     54 ;               Fix loop problem in cleaning up receives.  Return SS$_CONNECFAIL
0000     55 ;               when an IO$_ACCESS fct can't locate the XWB (was SS$_NOLINKS).
0000     56 ;               Send NET$C_DR_ABORT upon IO$_DEACCESS!IO$M_ABORT (was sending
0000     57 ;               NET$C_DR_NORMAL).
0000     58 ;
0000     59 ;       V03-019 LMP0221         L. Mark Pilant,         7-Apr-1984  14:29
0000     60 ;               Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000     61 ;               ORB$W_PROT.
0000     62 ;
0000     63 ;       V03-018 ADE1041         A. Eldridge      7-Mar-1984
0000     64 ;               Fix resource error count -- registers were screwed up.
0000     65 ;
0000     66 ;       V03-017 ADE1040         A.Eldridge      10-Sep-1983
0000     67 ;               Major rewrite to accomodate changes to allow NSP (NETDRVNSP.MAR)
0000     68 ;               to use kernel mode AST's to nibble away at the user buffers
0000     69 ;               rather than accessing them just at FDT or I/O post time.  This
0000     70 ;               change was needed to allow huge user buffers (for performance)
0000     71 ;               without requiring a lot of pool.
0000     72 ;
0000     73 ;
```

```
0000     75
0000     76 .SBTTL  DECLARATIONS
0000     77 ;
0000     78 ; INCLUDE FILES:
0000     79 ;
0000     80
0000     81          $AQBDEF
0000     82          $ACBDEF
0000     83          $CCBDEF
0000     84          $CRBDEF
0000     85          $CXBDEF
0000     86          $DDBDEF
0000     87          $DRDEF
0000     88          $DYNDEF
0000     89          $FKBDEF
0000     90          $IODEF
0000     91          $IPLDEF
0000     92          $IRPDEF
0000     93          $JIBDEF
0000     94          $MSGDEF
0000     95          $ORBDEF
0000     96          $PCBDEF
0000     97          $PHDDEF
0000     98          $PRDEF
0000     99          $SSDEF
0000    100          $TQEDEF
0000    101          $UCBDEF
0000    102          $VECDEF
0000    103
0000    104          $ICBDEF
0000    105          $IDBDEF
0000    106          $LPDDEF
0000    107          $LTBDEF
0000    108          $LLIDEF
0000    109          $RCBDEF
0000    110
0000    111          $NETSYMDEF
0000    112          $NETUPDDEF
0000    113          $NSPMSGDEF
0000    114
0000    115          $CXBEXTDEF              ; NETDRIVER CXB extensions
0000    116          $XWBDEF                 ; XWB and LSB definitions
0000    117
```

```
0000   119
0000   120 ;
0000   121 ; MACROS:
0000   122 ;
0000   123
0000   124 ;
0000   125 ;  Bit definition macro
0000   126 ;
0000   127 .MACRO  BITDEF  BLK,SYM,BITVAL
0000   128
0000   129         'BLK'$V_'SYM' = BITVAL
0000   130         'BLK'$M_'SYM' = 1@<BITVAL>
0000   131 .ENDM
0000   132
0000   133 ;
0000   134 ;  Macro to set up mailbox message filtering table
0000   135 ;
0000   136 .MACRO  MBX_FILTER      MESSAGE,BIT
0000   137
0000   138         .LONG   MBX$M_'BIT
0000   139         .WORD   MSG$_'MESSAGE
0000   140
0000   141 .ENDM   MBX_FILTER
0000   142
0000   143 ;
0000   144 ;  Macro to build a mask of XWB$M_FLG_xxx bits
0000   145 ;
0000   146 .MACRO  BLDMSK  A
0000   147         _$MSK   = _$MSK + XWB$M_FLG_'A'
0000   148 .ENDM
0000   149
0000   150 ;
0000   151 ;  Macro to fill the 'set' and 'clear' XWB$W_FLG tables
0000   152 ;
0000   153 .MACRO  STATEMASK       STA,SETM,CLRM
0000   154
0000   155         ;
0000   156         ;  Build and enter the 'set FLG' bit mask
0000   157         ;
0000   158         _$MSK = 0
0000   159         .IRP    A,<SETM>
0000   160                 BLDMSK  A
0000   161         .ENDR
0000   162         . = NET$AW_FLG_SETM + <2*XWB$C_STA_'STA'>
0000   163         .WORD   _$MSK
0000   164         ;
0000   165         ;  Build and enter the 'clear FLG' bit mask
0000   166         ;
0000   167         _$MSK = 0
0000   168         .IRP    A,<CLRM>
0000   169                 BLDMSK  A
0000   170         .ENDR
0000   171         . = NET$AW_FLG_CLRM + <2*XWB$C_STA_'STA'>
0000   172         .WORD   _$MSK
0000   173 .ENDM
0000   174
```

```
0000   176
0000   177 ;
0000   178 ;  Macro to initialize NSP state tables
0000   179 ;
0000   180 .MACRO  STTAB                                ; Init state transition data
0000   181
0000   182         _$EVENT_INDEX   =  0                 ; Init event index
0000   183         _$ACT_INDEX     =  0                 ; Init action routine index
0000   184         ACT$_BUG        == 0                 ; Init the "bug-check" action routine
0000   185                                              ; index
0000   186
0000   187         _$ACT_DFLT = <XWB$C_STA_CLO -        ; Default state table entry
0000   188                      @NET$C_ACTBITS> -       ;
0000   189                      + ACT$_BUG              ;
0000   190
0000   191
0000   192 NET$AW_FLG_SETM:                            ; Bits to be set upon entering state
0000   193                    .BLKW XWB$C_NUMSTA        ; and upon timeout
0000   194 NET$AW_FLG_CLRM:.BLKW XWB$C_NUMSTA          ; Bits to be cleared upon entering state
0000   195
0000   196 NET$AB_STTAB:   .BLKB   0                   ; Bind the table address
0000   197
0000   198 .ENDM
0000   199
0000   200 ;
0000   201 ;  Macroes to move the current position within the state table
0000   202 ;
0000   203 .MACRO  ENDSTTAB                                    ; Move PC to end of table
0000   204         . = NET$AB_STTAB -
0000   205             + <_$EVENT_INDEX * XWB$C_NUMSTA>
0000   206 .ENDM
0000   207
0000   208
0000   209 .MACRO  EVENT  EV                                   ; Setup for this event
0000   210
0000   211         EV ==  _$EVENT_INDEX                       ; Define event code
0000   212         .=NET$AB_STTAB + <EV * XWB$C_NUMSTA>       ; Move PC to proper event
0000   213         .BYTE   <_$ACT_DFLT>[XWB$C_NUMSTA]         ; Init the entry
0000   214                                                    ;
0000   215         .=NET$AB_STTAB + <EV * XWB$C_NUMSTA>       ; Move PC to proper event
0000   216         _$EVENT_INDEX  =  _$EVENT_INDEX + 1        ; Get ready for next event
0000   217
0000   218 .ENDM
0000   219
0000   220
0000   221 ;
0000   222 ;  Macro to fill the build and enter the state transition table element
0000   223 ;
0000   224 .MACRO  STATE   CURSTA,NXTSTA,ACTION,?LL           ; Make table entry
0000   225     LL:
0000   226         .=.+XWB$C_STA_'CURSTA'                     ; Goto state table entry
0000   227         ;
0000   228         ;
0000   229         ;   If the action routine index is not defined then define the index
0000   230
0000   231         ;   Create the state-table entry.
0000   232         ;
```

```
0000   233          ;
0000   234          .IF NDF,ACT$_'ACTION'
0000   235                  ACT$_'ACTION'    = _$ACT_INDEX
0000   236                  _$ACT_INDEX      = _$ACT_INDEX + 1
0000   237          .ENDC
0000   238          .BYTE   <XWB$C_STA_'NXTSTA' @NET$C_ACTBITS> + ACT$_'ACTION'
0000   239
0000   240      .=LL
0000   241  .ENDM
0000   242
```

```
                     0000   244
                     0000   245 ;
                     0000   246 ; EQUATED SYMBOLS:
                     0000   247 ;
                     0000   248
                     0000   249 ;
                     0000   250 ;  Argument list offsets for QIO
                     0000   251 ;
00000000             0000   252 P1      = 0                                 ; Buffer address
00000004             0000   253 P2      = 4                                 ; Buffer length
00000008             0000   254 P3      = 8                                 ; Miscellaneous
                     0000   255
                     0000   256
                     0000   257 ASSUME  FKB$C_LENGTH  LE  ACB$C_LENGTH
                     0000   258
00000120             0000   259  $tmp           == <XWB$S_XWB+7>&^C<7>      ; XWB length, quad word aligned
00000160             0000   260 XWB$$           == .$tmp+64                 ; Allow enough room for the route-header
00000120             0000   261 XWB$L_PTR_RTHD  == .$tmp                    ; Ptr to route-header
00000124             0000   262 XWB$B_ADJ_INX   == .$tmp +4                 ; Adjacency index
00000158             0000   263 XWB$T_TR3HDR    == XWB$$-8                  ; Start of standard Phase III header
                     0000   264                                             ;  (must be quadword aligned)
                     0000   265
0000017C             0000   266 XWB_C_LEN = XWB$$+ACB$C_LENGTH               ; Total XWB length
                     0000   267
                     0000   268 ;
                     0000   269 ;  Definitions for mailbox message filtering
                     0000   270 ;
                     0000   271 $VIELD MBX,0,<-
                     0000   272                 <NETSTATE,,M>,-             ; Network state change
                     0000   273                 <EVTAVL,,M>,-               ; Events available for logging
                     0000   274                 <EVTRCVCHG,,M>,-            ; Event receiver database change
                     0000   275                 <EVTXMTCHG,,M>,-            ; Event xmitter database change
                     0000   276 >
                     0000   277
                     0000   278 ;
                     0000   279 ;
                     0000   280 ;  Define a mask containing all bits indicating work needs to be done
                     0000   281 ;
                     0000   282 ;
                     0000   283 XWB$M_FLG_WMSK =         XWB$M_FLG_SCD ! -
                     0000   284                          XWB$M_FLG_SDT ! XWB$M_FLG_SDACK!-
                     0000   285                          XWB$M_FLG_SLI ! XWB$M_FLG_SIACK!-
0000039D             0000   286                          XWB$M_FLG_CLO ! XWB$M_FLG_BREAK
                     0000   287
```

```
                        ;
                        ; DRIVER PROLOGUE TABLE
                        ;

                                .PSECT  $$$105_PROLOGUE
                        DPTAB   -
                                END     = NET$END,-                     ; End of driver
                                ADAPTER = NULL,-                        ; Adapter type
                                UCBSIZE = UCB$C_LENGTH,-                ; UCB size
                                NAME    = NETDRIVER                     ; Driver name

                        DPT_STORE INIT                                  ; CONTROL BLOCK INIT VALUES
                                                                        ;
                        DPT_STORE CRB,CRB$L_INTD+VEC$L_ADP,L,0          ; No ADP pointer
                        DPT_STORE UCB,UCB$W_MB_SEED,W,0                 ; Init. unit value for cloning
                        DPT_STORE UCB,UCB$B_FIPL,B,NET$C_IPL            ; Fork IPL
                        DPT_STORE UCB,UCB$B_DIPL,B,NET$C_IPL            ; Device IPL
                        DPT_STORE ORB,ORB$B_FLAGS,B,-
                                                <ORB$M_PROT_16>         ; SOGW protection word
                        DPT_STORE ORB,ORB$W_PROT,W,0                    ; Default protection
                        DPT_STORE ORB,ORB$L_OWNER,L,<^X010001>          ; Owner UIC
                        DPT_STORE UCB,UCB$L_DEVCHAR,L,-                 ; Device characteristics
                                                <DEV$M_NET!-            ;   Network device
                                                 DEV$M_AVL!-            ;   Available
                                                 DEV$M_MBX!-            ;   Mailbox type (no hardware)
                                                 DEV$M_IDV!-            ;   Input device
                                                 DEV$M_ODV-             ;   Output device
                                                >                       ;
                        DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,256             ;
                        DPT_STORE UCB,UCB$L_DEVDEPEND,L,-               ;
                                                MBX$M_NETSTATE          ; Enable NETSHUT by default
                        DPT_STORE UCB,UCB$W_STS,W,-                     ;
                                                <UCB$M_ONLINE!-         ; Device online
                                                 UCB$M_TEMPLATE-        ; NET0 is the "template" UCB
                                                >                       ; used to build other NET UCBs
                                                                        ;
                        DPT_STORE REINIT                                ; CONTROL BLOCK RE-INIT VALUES

                        DPT_STORE DDB,DDB$L_DDT,D,NET$DDT
                        DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL, D,NET$CTLR_INIT
                        DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,NET$UNIT_INIT
                        DPT_STORE CRB,CRB$L_INTD+VEC$L_START,   D,NET$ACP_COMM
                        DPT_STORE CRB,CRB$L_INTD+4,             D,NET$INTERRUPT

                        DPT_STORE END                                   ;
```

```
                    338
                    339 ;
                    340 ; DRIVER DISPATCH TABLE
                    341 ;
00000000            342         .PSECT  $$$115_DRIVER,LONG
                    343
                    344         DDTAB   DEVNAM  = NET,-          ; DRIVER DISPATCH TABLE
                    345                 FUNCTB  = FUNCTABLE,-     ; Function decision table address
                    346                 START   = NET$STARTIO,-   ; Start I/O operation
                    347                 ALTSTART= NET$ALTENTRY,-; Alternate I/O request entry point
                    348                 CANCEL  = NET$CANCEL,-    ; Cancel I/O entry point
                    349                 UNSOLIC = NET$UNSOL_INTR; Unsolicited interrupt
                    350
                    351
                    352 .SBTTL  FUNCTION DECISION TABLE
                    353
                    354 FUNCTABLE:                               ; FUNCTION DECISION TABLE
                    355         FUNCTAB ,-                      ; Legal Functions
                    356                 <READVBLK,READLBLK,-    ; Read
                    357                  WRITEVBLK,WRITELBLK,-  ; Write
                    358                  SETMODE,-              ; Set mailbox message filters
                    359                  ACCESS,-               ; Logical-link Connect/Reject
                    360                  ACPCONTROL,-           ; ACP Control function
                    361                  DEACCESS,-             ; Logical-link Disconnect
                    362                 >
                    363         FUNCTAB ,-                      ; BUFFERED I/O FUNCTIONS
                    364                 <READVBLK,READLBLK,-    ; Read
                    365                  WRITEVBLK,WRITELBLK,-  ; Write
                    366                  SETMODE,-              ; Set mailbox message filters
                    367                  ACCESS,-               ; Logical-link Connect/Reject
                    368                  ACPCONTROL,-           ; ACP Control function
                    369                  DEACCESS,-             ; Logical-link Disconnect
                    370                 >
0048                371         FUNCTAB NET$FDT_RCV,     <READLBLK>    ; Read
0054                372         FUNCTAB NET$FDT_XMT,     <WRITELBLK>   ; Write
0060                373         FUNCTAB NET$FDT_ACCESS,  <ACCESS>      ; Connect Logical-link
006C                374         FUNCTAB NET$FDT_DEACCESS,<DEACCESS>    ; Disconnect Logical-link
0078                375         FUNCTAB NET$FDT_SETMODE, <SETMODE>     ; Set mailbox message filters
0084                376         FUNCTAB NET$FDT_CONTROL, <ACPCONTROL>  ; ACP Control
```

```
                     0090   378 .SBTTL  State Table
                     0090   379
                     0090   380 ;
                     0090   381 ;  OWN STORAGE:
                     0090   382 ;
00000080             0090   383         PATCH_AREA_SIZE = 128           ; Size of patch area space
                     0090   384
                     0090   385 NET$GQ_PATCH::
00000080             0090   386         .LONG   PATCH_AREA_SIZE
00000098'            0094   387         .LONG   .+4                     ; (not an address - offset from start
                     0098   388                                         ; of image to base of patch space)
00000118             0098   389         .BLKB   PATCH_AREA_SIZE
                     0118   390
                     0118   391
FFFFFEF5'            0118   392 NET$GL_OFF_DPTFLG::  .LONG  DPT$TAB + DPT$B_FLAGS - . ; Offset to DPT$B_FLAGS
                     011C   393
00000005             011C   394         NET$C_ACTBITS   = 5              ; Number of action bits per entry
00000003             011C   395         NET$C_STABITS   = 3              ; Number of state bits per entry
000000E0             011C   396         NET$M_STAMSK    = <7>@5          ; State bit mask
                     011C   397
                     011C   398 ;
                     011C   399 ;
                     011C   400 ;  The following definitions must be contiguous to the NSPTABLES definition
                     011C   401 ;
                     011C   402 ;
                     011C   403 STTAB                                    ; Init state transition table
                     013C   404
                     013C   405 EVENT      NETEVT$_CI                    ; CI message received
                     013C   406     STATE  CIS, CIS,    BUG              ; something wrong in the driver
                     013C   407     STATE  CAR, CAR,    LOG              ; unexpected event
                     013C   408     STATE  CIR, CIR,    RCV_CR           ; Assume received retransmitted CI
                     013C   409     STATE  CCS, CCS,    RCV_CR           ; Assume received retransmitted CI
                     013C   410     STATE  RUN, RUN,    LOG              ; unexpected event
                     013C   411     STATE  DIS, DIS,    LOG              ; unexpected event
                     013C   412     STATE  DIR, DIR,    LOG              ; unexpected event
                     013C   413     STATE  CLO, CIR,    RCV_CI           ; inbound connect sequence
                     013C   414
                     013C   415 EVENT      NETEVT$_CA                    ; Connect Ack received
                     0144   416     STATE  CIS, CAR,    RCV_CA           ; measure intial round-trip time
                     0144   417     STATE  CAR, CAR,    NOP              ; assume retransmission
                     0144   418     STATE  CIR, CIR,    LOG              ; unexpected event
                     0144   419     STATE  CCS, CCS,    LOG              ; unexpected event
                     0144   420     STATE  RUN, RUN,    NOP              ; assume late arrival
                     0144   421     STATE  DIS, DIS,    NOP              ; assume late arrival
                     0144   422     STATE  DIR, DIR,    NOP              ; assume late arrival
                     0144   423     STATE  CLO, CLO,    NOP              ; assume late arrival
                     0144   424
                     0144   425 EVENT      NETEVT$_CC                    ; Connect Confirm received
                     014C   426     STATE  CIS, RUN,    RCV_CC           ; normal handshaking sequence
                     014C   427     STATE  CAR, RUN,    RCV_CC           ; normal handshaking sequence
                     014C   428     STATE  CIR, CIR,    LOG              ; unexpected event
                     014C   429     STATE  CCS, CCS,    LOG              ; unexpected event
                     014C   430     STATE  RUN, RUN,    NOP              ; assume retransmission
                     014C   431     STATE  DIS, DIS,    NOP              ; we enter DIS for many reasons
                     014C   432     STATE  DIR, DIR,    NOP              ; assume late arrival
                     014C   433     STATE  CLO, CLO,    RTS_NLT          ; assume late arrival
                     014C   434
```

```
014C    436
014C    437 EVENT       NETEVT$_PH2CCS                  ; Phase II connect confirm xmt-complete
0154    438     STATE   CIS, CIS,       NOP             ;
0154    439     STATE   CAR, CAR,       NOP             ;
0154    440     STATE   CIR, CIR,       NOP             ;
0154    441     STATE   CCS, RUN,       ENT_RUN         ; Normal Phase II handshaking sequence
0154    442     STATE   RUN, RUN,       NOP             ;
0154    443     STATE   DIS, DIS,       NOP             ;
0154    444     STATE   DIR, DIR,       NOP             ;
0154    445     STATE   CLO, CLO,       NOP             ;
0154    446
0154    447 EVENT       NETEVT$_RTS                     ; Rcv "return to sender" CI message
015C    448     STATE   CIS, CLO,       RCV_RTS         ; Process returned message
015C    449     STATE   CAR, CAR,       NOP             ; Assume late arrival on retransmission
015C    450     STATE   CIR, CIR,       NOP             ; Assume late arrival on retransmission
015C    451     STATE   CCS, CCS,       NOP             ; Assume late arrival on retransmission
015C    452     STATE   RUN, RUN,       NOP             ; Assume late arrival on retransmission
015C    453     STATE   DIS, DIS,       NOP             ; Assume late arrival on retransmission
015C    454     STATE   DIR, DIR,       NOP             ; Assume late arrival on retransmission
015C    455     STATE   CLO, CIR,       NOP             ; Assume late arrival on retransmission
015C    456
015C    457 EVENT       NETEVT$_DATA                    ; Data message received
0164    458     STATE   CIS, CIS,       LOG             ; unexpected event
0164    459     STATE   CAR, CAR,       LOG             ; unexpected event
0164    460     STATE   CIR, CIR,       LOG             ; unexpected event
0164    461     STATE   CCS, RUN,       ENT_RUN         ; a normal handshaking sequence
0164    462     STATE   RUN, RUN,       RCV_DATA        ; this is what NSP is for
0164    463     STATE   DIS, DIS,       NOP             ; unavoidable race in sending DI
0164    464     STATE   DIR, DIR,       NOP             ; assume late arrival
0164    465     STATE   CLO, CLO,       RTS_NLT         ; assume late arrival
0164    466
0164    467 EVENT       NETEVT$_DTACK                   ; Data Ack received
016C    468     STATE   CIS, CIS,       LOG             ; unexpected event
016C    469     STATE   CAR, CAR,       LOG             ; unexpected event
016C    470     STATE   CIR, CIR,       LOG             ; unexpected event
016C    471     STATE   CCS, RUN,       ENT_RUN         ; a normal handshaking sequence
016C    472     STATE   RUN, RUN,       RCV_DTACK       ; drive the link
016C    473     STATE   DIS, DIS,       NOP             ; assume late arrival or race
016C    474     STATE   DIR, DIR,       NOP             ; assume late arrival or race
016C    475     STATE   CLO, CLO,       RTS_NLT         ; assume late arrival or race
016C    476
```

```
016C    478
016C    479 EVENT   NETEVT$_LS                      ; Link Service msg received
0174    480     STATE   CIS, CIS,   LOG             ; unexpected event
0174    481     STATE   CAR, CAR,   LOG             ; unexpected event
0174    482     STATE   CIR, CIR,   LOG             ; unexpected event
0174    483     STATE   CCS, RUN,   ENT_RUN         ; a normal handshaking sequence
0174    484     STATE   RUN, RUN,   RCV_LI          ; drive the link
0174    485     STATE   DIS, DIS,   NOP             ; assume late arrival or race
0174    486     STATE   DIR, DIR,   NOP             ; assume late arrival or race
0174    487     STATE   CLO, CLO,   RTS_NLT         ; assume late arrival or race
0174    488
0174    489 EVENT   NETEVT$_INT                     ; Interrupt msg received
017C    490     STATE   CIS, CIS,   LOG             ; unexpected event
017C    491     STATE   CAR, CAR,   LOG             ; unexpected event
017C    492     STATE   CIR, CIR,   LOG             ; unexpected event
017C    493     STATE   CCS, RUN,   ENT_RUN         ; a normal handshaking sequence
017C    494     STATE   RUN, RUN,   RCV_LI          ; drive the link
017C    495     STATE   DIS, DIS,   NOP             ; assume late arrival or race
017C    496     STATE   DIR, DIR,   NOP             ; assume late arrival or race
017C    497     STATE   CLO, CLO,   RTS_NLT         ; assume late arrival or race
017C    498
017C    499 EVENT   NETEVT$_LIACK                   ; INT/LS Ack received
0184    500     STATE   CIS, CIS,   LOG             ; unexpected event
0184    501     STATE   CAR, CAR,   LOG             ; unexpected event
0184    502     STATE   CIR, CIR,   LOG             ; unexpected event
0184    503     STATE   CCS, RUN,   ENT_RUN         ; a normal handshaking sequence
0184    504     STATE   RUN, RUN,   RCV_LIACK       ; drive the link
0184    505     STATE   DIS, DIS,   NOP             ; assume late arrival or race
0184    506     STATE   DIR, DIR,   NOP             ; assume late arrival or race
0184    507     STATE   CLO, CLO,   RTS_NLT         ; assume late arrival or race
0184    508
0184    509 EVENT   NETEVT$_DI                      ; Disconnect Initiate msg rcv'd
018C    510     STATE   CIS, DIR,   RCV_Dx          ; link rejected
018C    511     STATE   CAR, DIR,   RCV_Dx          ; link rejected
018C    512     STATE   CIR, DIR,   ABORT           ; abort the link, no local owner
018C    513     STATE   CCS, DIR,   RCV_Dx          ; abort the link
018C    514     STATE   RUN, DIR,   RCV_Dx          ; abort the link
018C    515     STATE   DIS, DIR,   ABORT           ; change state and send DC
018C    516     STATE   DIR, DIR,   NOP             ; send DC
018C    517     STATE   CLO, CLO,   RTS_NLT         ; assume race or late arrival
018C    518
018C    519 EVENT   NETEVT$_DC                      ; Disconnect Confirm msg rcv'd
0194    520     STATE   CIS, CLO,   RCV_Dx          ; link rejected
0194    521     STATE   CAR, CLO,   RCV_Dx          ; link rejected
0194    522     STATE   CIR, CLO,   ABORT           ; link aborted, no local owner
0194    523     STATE   CCS, CLO,   RCV_Dx          ; link aborted
0194    524     STATE   RUN, CLO,   RCV_Dx          ; link aborted
0194    525     STATE   DIS, CLO,   NOP             ; normal handshaking sequence
0194    526     STATE   DIR, CLO,   NOP             ; assume DC is a 'no link terminate'
0194    527     STATE   CLO, CLO,   NOP             ; assume late arrival
0194    528
0194    529
```

```
0194    531
0194    532 EVENT       NETEVT$_DSCLNK                  ; Link failed confidence test
019C    533     STATE   CIS, CLO,      ABORT           ; connect timed out
019C    534     STATE   CAR, DIS,      ABORT           ; connect timed out
019C    535     STATE   CIR, DIS,      ABORT           ; local system is slow
019C    536     STATE   CCS, DIS,      ABORT           ; connect timed out
019C    537     STATE   RUN, DIS,      ABORT           ; problem talking with remote node
019C    538     STATE   DIS, CLO,      NOP             ; abort the link
019C    539     STATE   DIR, CLO,      NOP             ; abort the link
019C    540     STATE   CLO, CLO,      NOP             ; Try to deallocate XWB
019C    541
019C    542
019C    543 EVENT       NETEVT$_CANLNK                  ; Local cancel of link
01A4    544     STATE   CIS, CLO,      CANLNK          ; abort from a Connect state
01A4    545     STATE   CAR, CLO,      CANLNK          ; abort from a Connect state
01A4    546     STATE   CIR, DIS,      CANLNK          ; abort link, no local owner
01A4    547     STATE   CCS, DIS,      CANLNK          ; abort from a Connect state
01A4    548     STATE   RUN, DIS,      CANLNK          ; orderly shuttdown
01A4    549     STATE   DIS, DIS,      NOP             ; link is already disconnecting
01A4    550     STATE   DIR, DIR,      NOP             ; link is already disconnecting
01A4    551     STATE   CLO, CLO,      NOP             ; link is already disconnecting
01A4    552
01A4    553 EVENT       NETEVT$_RESDIS                  ; Resume disconnect
01AC    554     STATE   CIS, CIS,      BUG             ; Valid only from RUN state
01AC    555     STATE   CAR, CAR,      BUG             ; Valid only from RUN state
01AC    556     STATE   CIR, CIR,      BUG             ; Valid only from RUN state
01AC    557     STATE   CCS, CCS,      BUG             ; Valid only from RUN state
01AC    558     STATE   RUN, DIS,      RES_DISC        ; Disconnect if XWB is idle
01AC    559     STATE   DIS, DIS,      BUG             ; Valid only from RUN state
01AC    560     STATE   DIR, DIR,      BUG             ; Valid only from RUN state
01AC    561     STATE   CLO, CLO,      BUG             ; Valid only from RUN state
01AC    562
01AC    563
```

```
01AC    565
01AC    566 EVENT       NETEVT$_CIA                     ; Connect Initiate IO$_ACCESS
01B4    567     STATE   CIS, CIS,       BUG             ; XWB was just created
01B4    568     STATE   CAR, CAR,       BUG             ; XWB was just created
01B4    569     STATE   CIR, CAR,       BUG             ; XWB was just created
01B4    570     STATE   CCS, CCS,       BUG             ; XWB was just created
01B4    571     STATE   RUN, RUN,       BUG             ; XWB was just created
01B4    572     STATE   DIS, DIS,       BUG             ; XWB was just created
01B4    573     STATE   DIR, DIR,       BUG             ; XWB was just created
01B4    574     STATE   CLO, CIS        INITIATE        ; Normal connect initiate seq.
01B4    575
01B4    576 EVENT       NETEVT$_CCA                     ; Connect Confirm IO$_ACCESS
01BC    577     STATE   CIS, CIS,       SSABORT         ; Confirm not possible
01BC    578     STATE   CAR, CAR,       SSABORT         ; Confirm not possible
01BC    579     STATE   CIR, CCS,       CONFIRM         ; Normal connect confirm seq.
01BC    580     STATE   CCS, CCS,       SSABORT         ; Confirm not possible
01BC    581     STATE   RUN, RUN,       SHRLNK          ; Second accessor to link
01BC    582     STATE   DIS, DIS,       SSABORT         ; Confirm no longer possible
01BC    583     STATE   DIR, DIR,       SSABORT         ; Confirm no longer possible
01BC    584     STATE   CLO, CLO,       SSABORT         ; Confirm no longer possible
01BC    585
01BC    586 EVENT       NETEVT$_CRA                     ; Connect Reject IO$_ACCESS
01C4    587     STATE   CIS, CIS,       SSABORT         ; Reject not possible
01C4    588     STATE   CAR, CAR,       SSABORT         ; Reject not possible
01C4    589     STATE   CIR, DIS,       CONFIRM         ; Normal connect reject seq.
01C4    590     STATE   CCS, CCS,       SSABORT         ; Reject not possible
01C4    591     STATE   RUN, RUN,       SSABORT         ; Reject not possible
01C4    592     STATE   DIS, DIS,       SSABORT         ; Reject not possible
01C4    593     STATE   DIR, DIR,       SSABORT         ; Reject not possible
01C4    594     STATE   CLO, CLO,       SSABORT         ; Reject not possible
01C4    595
01C4    596 EVENT       NETEVT$_DEA                     ; QIO IO$_DEACCESS
01CC    597     STATE   CIS, CIS,       BUG             ; Channel should not have window
01CC    598     STATE   CAR, CAR,       BUG             ; Channel should not have window
01CC    599     STATE   CIR, CIR,       BUG             ; Channel should not have window
01CC    600     STATE   CCS, CCS,       BUG             ; Channel should not have window
01CC    601     STATE   RUN, DIS,       DEACCESS        ; But change to DIS state only
01CC    602                                             ;  if this is the last accessor
01CC    603     STATE   DIS, DIS,       DEACCESS        ; Link was aborted externally
01CC    604     STATE   DIR, DIR,       DEACCESS        ; Link was aborted externally
01CC    605     STATE   CLO, CLO,       DEACCESS        ; Link was aborted externally
01CC    606
```

```
01CC    608
01CC    609 EVENT       NETEVT$_MBXERR                  ; Fatal error writing to mailbox
01D4    610     STATE   CIS, CLO,      ABORT           ; abort from a Connect state
01D4    611     STATE   CAR, CLO,      ABORT           ; abort from a Connect state
01D4    612     STATE   CIR, DIS,      ABORT           ; abort link, no local owner
01D4    613     STATE   CCS, DIS,      ABORT           ; abort from a Connect state
01D4    614     STATE   RUN, DIS,      ABORT           ; abort from the RUN state
01D4    615     STATE   DIS, DIS,      NOP             ; link is already disconnecting
01D4    616     STATE   DIR, DIR,      NOP             ; link is already disconnecting
01D4    617     STATE   CLO, CLO,      NOP             ; link is already disconnecting
01D4    618
01D4    619 EVENT       NETEVT$_PROERR                  ; Protocol error     (NOP for now)
01DC    620     STATE   CIS, CIS,      NOP             ;
01DC    621     STATE   CAR, CAR,      NOP             ;
01DC    622     STATE   CIR, DIS,      NOP             ;
01DC    623     STATE   CCS, DIS,      NOP             ;
01DC    624     STATE   RUN, RUN,      NOP             ;
01DC    625     STATE   DIS, DIS,      NOP             ;
01DC    626     STATE   DIR, DIR,      NOP             ;
01DC    627     STATE   CLO, CLO,      NOP             ;
01DC    628
01DC    629
```

```
                01DC    631
                01DC    632 ;
                01DC    633 ;  Setup tables which specify which XWB$W_FLG bits to set and clear upon
                01DC    634 ;  a transition into a new state.
                01DC    635 ;
                01DC    636 ;          New
                01DC    637 ;          State  Flags to set     Flags to clear
                01DC    638 ;          -----  ------------     --------------
                01DC    639 ;
                01DC    640 STATEMASK CIS,  <SCD>             <WBUF>
                0130    641 STATEMASK CAR,  <CLO>             <WBUF>
                0132    642 STATEMASK CIR,  <SCD>             <WBUF>
                0134    643 STATEMASK CCS,  <SCD>             <WBUF>
                0136    644 STATEMASK RUN,  <SDT,SDFL,WHGL>   <WBUF,SCD>
                0138    645 STATEMASK DIR,  <SCD>             <WBUF,WBP,WHGL,WDAT,SDT,SLI,SDACK,SIACK,-
                0138    646                                                 BREAK,IAVL,TBPR,SIFL,SDFL>
                013A    647 STATEMASK DIS,  <SCD>             <WBUF,WBP,WHGL,WDAT,SDT,SLI,SDACK,SIACK,-
                013A    648                                                 BREAK,IAVL,TBPR,SIFL,SDFL>
                013C    649 STATEMASK CLO,  <CLO>             <WBUF,WBP,WHGL,WDAT,SDT,SLI,SDACK,SIACK,-
                013C    650                                               SCD,BREAK,IAVL,TBPR,SIFL,SDFL>
                012E    651
                012E    652 ENDSTTAB
                01E4    653
                01E4    654
                01E4    655 ;
                01E4    656 ; The following mask is used to identify the subset of flags used
                01E4    657 ; to signal work to be done
                01E4    658 ;
0000039D        01E4    659 NET$GL_WORKBITS:: .LONG XWB$M_FLG_WMSK  ; Flags requiring work to be done
                01E8    660
                01E8    661
                01E8    662 MBX_TABLE:                                      ; Table for mapping mbx msg codes
                01E8    663                                                 ; to filter bits
                01E8    664         MBX_FILTER      NETSHUT,NETSTATE        ; Network state change
                01EE    665         MBX_FILTER      EVTAVL,EVTAVL           ; Events available for logging
                01F4    666         MBX_FILTER      EVTRCVCHG,EVTRCVCHG     ; Event receiver database change
                01FA    667         MBX_FILTER      EVTXMTCHG,EVTXMTCHG     ; Event xmitter database change
00000000        0200    668         .LONG   0                               ; End of table
                0204    669
```

```
                     0204   671 .SBTTL  NET$AZ_DR_TABLE - Disconnect Reason Code Mapping
                     0204   672
                     0204   673
                     0204   674 ;
                     0204   675 ;  Macro to set up connect reject reason codes
                     0204   676 ;
00000000             0204   677 REASON_W_DR      == 0             ; Reason code
00000002             0204   678 REASON_W_SS      == 2             ; SS$_.. to return in data IRP's
00000004             0204   679 REASON_W_MBX     == 4             ; MBX$_.. message code
00000006             0204   680 REASON_C_LENGTH == 6
                     0204   681
                     0204   682 .MACRO  MRC     REASON,SS_CODE,MSG_CODE
                     0204   683
                     0204   684         .WORD   NET$C_DR_'REASON
                     0204   685         .WORD   SS$_'SS_CODE
                     0204   686         .WORD   MSG$_'MSG_CODE
                     0204   687
                     0204   688 .ENDM   MRC
                     0204   689
                     0204   690
00000064             0204   691 NET$C_DR_INVALID   == 100              ; Fake value meaning "not setup"
00000066             0204   692 NET$C_DR_DEACC     == 102              ; Fake value for code conversion
                     0204   693
                     0204   694 NET$AZ_DR_TABLE:                      ; Table for mapping disconnect reasons
                     0204   695                                       ; for other than the "connect-initiate"
                     0204   696                                       ; state
                     0204   697 ;
                     0204   698 ;
                     0204   699 ;    discon  data        mailbox
                     0204   700 ;    reason  status      message
                     0204   701 ;    ------  ----------  --------
                     0204   702 ;
                     0204   703 MRC  NORMAL, LINKDISCON, DISCON          ;
                     020A   704 MRC  EXIT,   LINKEXIT,   EXIT            ; User exit or timeout
                     0210   705 MRC  NOPATH, PATHLOST,   PATHLOST        ; Path lost to partner node
                     0216   706 MRC  SHUT,   SHUT,       NETSHUT         ; Node shutting down
                     021C   707 MRC  NOBJ,   PROTOCOL,   ABORT           ; No such object
                     0222   708 MRC  ABORT,  LINKABORT,  ABORT           ; Disconnect abort
                     0228   709 MRC  THIRD,  THIRDPARTY, THIRDPARTY      ; Disconnect by third party
                     022E   710 MRC  ACCESS, PROTOCOL,   ABORT           ; Login info invalid
                     0234   711 MRC  RSU,    PROTOCOL,   ABORT           ; Resource error
                     023A   712 MRC  BUSY,   PROTOCOL,   ABORT           ; Object too busy
                     0240   713 MRC  FMT,    PROTOCOL,   ABORT           ; Illegal process name field
                     0246   714 MRC  NONODE, PROTOCOL,   ABORT           ; Unrecognized node i.d.
                     024C   715 MRC  IVNODE, PROTOCOL,   ABORT           ; Invalid node-i.d. format
                     0252   716         ;
                     0252   717         ;
                     0252   718         ;  The following are internal codes and are not part of NSP
                     0252   719         ;
                     0252   720         ;
                     0252   721 MRC  DEACC,  LINKDISCON, DISCON          ; Link is IO$_DEACCESS'ed
                     0258   722 MRC  INVALID,LINKABORT,  ABORT           ; Reason field never setup
                     025E   723
FFFFFFFF             025E   724 .LONG -1                                ; Terminate the table (the last entry
                     0262   725                                        ; is to be used as a catch-all)
                     0262   726
                     0262   727
```

```
                     0262   728
                     0262   729 NET$AZ_DR_CONTAB:                       ; Table for mapping reject reasons
                     0262   730                                         ; in one of the "connect" states
                     0262   731 ;
                     0262   732 ;
                     0262   733 ;    discon  connect      mailbox
                     0262   734 ;    reason  status       message
                     0262   735 ;    ------  -----------  --------
                     0262   736 ;
                     0262   737 MRC  NORMAL, REJECT,      REJECT      ; Connect reject
                     0268   738 MRC  EXIT,   LINKEXIT,    EXIT        ; User exit or timeout
                     026E   739 MRC  NOPATH, UNREACHABLE, PATHLOST     ; Path lost to partner node
                     0274   740 MRC  SHUT,   SHUT,        NETSHUT     ; Node shutting down
                     027A   741 MRC  NOBJ,   NOSUCHOBJ,   REJECT      ; No such object
                     0280   742 MRC  ABORT,  LINKABORT,   ABORT       ; Disconnect abort
                     0286   743 MRC  THIRD,  THIRDPARTY,  THIRDPARTY  ; Disconnect by third party
                     028C   744 MRC  ACCESS, INVLOGIN,    REJECT      ; Login info invalid
                     0292   745 MRC  RSU,    REMRSRC,     REJECT      ; Resource error
                     0298   746 MRC  BUSY,   REMRSRC,     REJECT      ; Object too busy
                     029E   747 MRC  FMT,    PROTOCOL,    REJECT      ; Illegal process name field
                     02A4   748 MRC  NONODE, NOSUCHNODE,  REJECT      ; Unrecognized node i.d.
                     02AA   749 MRC  IVNODE, NOSUCHNODE,  REJECT      ; Invalid node-i.d. format
                     02B0   750      ;
                     02B0   751      ;
                     02B0   752      ;  The following are internal codes and are not part of NSP
                     02B0   753      ;
                     02B0   754      ;
                     02B0   755 MRC  DEACC,  ABORT,       ABORT       ; Link is IO$_DEACCESS'ed
                     02B6   756 MRC  INVALID,CONNECFAIL,  ABORT       ; Reason field never setup
                     02BC   757
                FFFFFFFF  02BC   758 .LONG -1                            ; Terminate the table (the last entry
                     02C0   759                                         ; is to be used as a catch-all)
                     02C0   760
                     02C0   761
                     02C0   762
                     02C0   763 NET$MAP_R_REASON::                       ; Map Reason code in XWB$W_R_REASON
50   FF3A CF    9E   02C0   764          MOVAB   W^NET$AZ_DR_TABLE -     ; Setup non-connect table address
                     02C5   765                   -REASON_C_LENGTH,R0   ; ...biased for scan
                     02C5   766
                     02C5   767          ASSUME  XWB$C_STA_CLO  EQ  0
                     02C5   768          ASSUME  XWB$C_STA_CIS  EQ  1
                     02C5   769          ASSUME  XWB$C_STA_CAR  EQ  2
                     02C5   770
  1E A5    02    91   02C5   771          CMPB    #2,XWB$B_STA(R5)        ; Is 'connect intiate' table needed?
           05    19   02C9   772          BLSS    10$                     ; If LSS then no
50   FF8D CF    9E   02CB   773          MOVAB   W^NET$AZ_DR_CONTAB -    ; Setup connect-intiate table address
                     02D0   774                   -REASON_C_LENGTH,R0   ; ...biased for scan
                     02D0   775                                         ;
     50    06    C0   02D0   776 10$:     ADDL    #REASON_C_LENGTH,R0     ; Goto next entry
       44  A5    B1   02D3   777          CMPW    XWB$W_R_REASON(R5),-    ; Does it match ?
                 60   02D6   778                  REASON_R_DR(R0)        ;
           07    13   02D7   779          BEQL    20$                     ; If EQL then yes
           60    D5   02D9   780          TSTL    (R0)                    ; At end of table?
           F3    18   02DB   781          BGEQ    10$                     ; If GEQU then no
     50    06    C2   02DD   782          SUBL    #REASON_C_LENGTH,R0     ; No match found, use the default entry
                 05   02E0   783 20$:     RSB
                      02E1   784
```

```
                02E1    785
                02E1    786
```

```
                02E1    788
                02E1    789 NET$INTERRUPT:
                02E1    790 NET$CTLR_INIT:
 05             02E1    791             RSB
                02E2    792 NET$UNIT_INIT:
 01             02E2    793             NOP
 01             02E3    794             NOP
 01             02E4    795             NOP
 01             02E5    796             NOP
 01             02E6    797             NOP
 01             02E7    798             NOP
 01             02E8    799             NOP
 01             02E9    800             NOP
 01             02EA    801             NOP
 05             02EB    802             RSB
                02EC    803
```

```
                     02EC    805 .SBTTL  NET$FORK        - Fork the XWB to do new work
                     02EC    806 ;+
                     02EC    807 ;
                     02EC    808 ;  If the fork block in the XWB is available, it is forked so that the work
                     02EC    809 ;  in XWB$W_FLG will be done.  If the fork block is unavailable, no further
                     02EC    810 ;  action is required since the XWB$W_FLG work will get done when to XWB fork
                     02EC    811 ;  process is subsequently resumed.
                     02EC    812 ;
                     02EC    813 ;
                     02EC    814 ;  INPUTS:       R5      XWB address
                     02EC    815 ;                 R0      Garbage
                     02EC    816 ;
                     02EC    817 ;  OUTPUTS:      R0      #1
                     02EC    818 ;
                     02EC    819 ;                 All other registers are preserved
                     02EC    820 ;
                     02EC    821 ;-
                     02EC    822 NET$FORK::                                          ; Fork the XWB
06 0E A5    02   E2  02EC    823          BBSS   #XWB$V_STS_SOL,XWB$W_STS(R5),20$ ; If BS, fork block in use
                     02F1    824                                                    ;
             3E   BB 02F1    825          PUSHR  #^M<R1,R2,R3,R4,R5>              ; Save regs
             06   10 02F3    826          BSBB    30$                             ; Schedule fork and return
             3E   BA 02F5    827 10$:     POPR   #^M<R1,R2,R3,R4,R5>             ;
                     02F7    828                                                    ;
          50 01   D0 02F7    829 20$:     MOVL   #1,R0                            ; Always return success
                  05 02FA    830          RSB                                     ; Done
                     02FB    831
       55 14 A5   9E 02FB    832 30$:     MOVAB   XWB$Q_FORK(R5),R5               ; Switch to fork block context
 00000000'GF      16 02FF    833          JSB     G^EXE$FORK                      ; Fork
       55 EC A5   9E 0305    834          MOVAB  -XWB$Q_FORK(R5),R5               ; Restore XWB context
                     0309    835                                                    ;
    0E A5    04   AA 0309    836          BICW   #XWB$M_STS_SOL,XWB$W_STS(R5)     ; We're back
    1C A5    02   AA 030D    837          BICW   #XWB$M_FLG_WBUF,XWB$W_FLG(R5)    ; Clear wait flag to allow retry
10 0E A5    03   E1 0311    838          BBC    #XWB$V_STS_DIS,XWB$W_STS(R5),100$; If BC, disconnect not pending
                     0316    839                                                    ;
          0FC0 8F   BB 0316    840          PUSHR  #^M<R6,R7,R8,R9,R10,R11>         ; Save Event regs
               5B   D4 031A    841          CLRL   R11                              ; Say "not okay to go to IPL 2"
          57   0E   9A 031C    842          MOVZBL #NETEVT$_RESDIS,R7               ; Event is "resume deaccess"
             0034   30 031F    843          BSBW   NET$EVENT                        ; Signal the event
          0FC0 8F   BA 0322    844          POPR   #^M<R6,R7,R8,R9,R10,R11>         ; Restore regs
                     0326    845                                                    ;
              007A   30 0326    846 100$:    BSBW   NET$SCH_MSG                      ; Schedule message transmission
                     0329    847          RSB                                     ; Done
                     032A    848
                     032A    849
```

```
                     032A   851 .SBTTL  NET$END_EVENT   - Abort current event without changing state
                     032A   852 .SBTTL  NET$COMPLEX_EV  - Change state and process new event
                     032A   853 .SBTTL  NET$PRE_EMPT    - Process new event without changing state
                     032A   854 ;+
                     032A   855 ;
                     032A   856 ;  These routines are called by the dispatched event action routines in order
                     032A   857 ;  to complete current event processing in a non-standard way.  They should be
                     032A   858 ;  considered substitutes to the RSB instruction which is normally used to
                     032A   859 ;  return control -- consequently the stack is checked for the return address
                     032A   860 ;  of the event dispatcher.
                     032A   861 ;
                     032A   862 ;  CALLING SEQUENCE:   JMP  NET$xxx
                     032A   863 ;
                     032A   864 ;  INPUTS: R10    Preserved for call to action routine
                     032A   865 ;          R9     The value originally stored by the event dispatcher
                     032A   866 ;          R8     Preserved for call to action routine
                     032A   867 ;          R7     Code of new event to be processed (scratch if NET$END_EVENT)
                     032A   868 ;          R6     The value originally stored by the event dispatcher
                     032A   869 ;          R5     XWB address
                     032A   870 ;          R4-R1  Scratch
                     032A   871 ;          R0     If NET$END_EVENT the status to be returned to the
                     032A   872 ;                    caller of the event dispatcher,
                     032A   873 ;                 Else scratch
                     032A   874 ;         (SP)    The address of CHANGE_STA which is the NET$EVENT return
                     032A   875 ;                 address.
                     032A   876 ;
                     032A   877 ;  OUTPUTS:  N/A
                     032A   878 ;
                     032A   879 ;-
                     032A   880 NET$END_EVENT::                          ; End event without changing state
           19   10   032A   881         BSBB    CHKRETADDR               ; Make sure stack is setup properly
         0074   30   032C   882         BSBW    NET$SCH_MSG              ; Schedule message transmission
                05   032F   883         RSB
                     0330   884
                     0330   885 NET$COMPLEX_EV::                         ; Change state, process new event
           13   10   0330   886         BSBB    CHKRETADDR               ; Validate state of stack
           05   EF   0332   887         EXTZV   #NET$C_ACTBITS,-         ;
                     0334   888                 #NET$C_STABITS,R9,R4     ; Get next state
      54   59   03         
        1E A5   54   91   0337   889         CMPB    R4,XWB$B_STA(R5)         ; New state ?
           02   13   033B   890         BEQL    10$                      ; If not, branch
           3F   10   033D   891         BSBB    NEW_STATE                ; Enter new state
           15   11   033F   892 10$:     BRB     NET$EVENT                ; Process new event
                     0341   893                                          ;
                     0341   894 NET$PRE_EMPT::                           ; Pre-empt the current event
           02   10   0341   895         BSBB    CHKRETADDR               ; Validate state of stack
           11   11   0343   896         BRB     NET$EVENT                ; Process new event
                     0345   897
                     0345   898 CHKRETADDR:                              ; Checks return address to trap bugs
         6F'AF   9F   0345   899         PUSHAB  B^CHANGE_STA             ; Prepare for next instruction
      04 AE   8E   D1   0348   900         CMPL    (SP)+,4(SP)              ; Is state of stack correct?
           04   12   034C   901         BNEQ    5$                       ; If NEQ then no
        6E   8E   D0   034E   902         MOVL    (SP)+,(SP)               ; Overlay return address
                05   0351   903         RSB                              ; Return
                     0352   904
                     0352   905 5$:      BUG_CHECK NETNOSTATE,FATAL
                     0356   906
```

```
                                0356   908 .SBTTL  NET$EVENT       - Event dispatcher
                                0356   909 ;+
                                0356   910 ;
                                0356   911 ;  This is the state table event dispatcher used to determine what is to be
                                0356   912 ;  done and what state the XWB is to enter next.  An event only has meaning
                                0356   913 ;  within the context of a XWB.
                                0356   914 ;
                                0356   915 ;
                                0356   916 ;  INPUTS:      R10     Preserved for call to action routine
                                0356   917 ;               R9      Available for the event dispatcher's exclusive use
                                0356   918 ;               R8      Preserved for call to action routine
                                0356   919 ;               R7      Code of the event to be processed
                                0356   920 ;               R6      If received message event then Tranport's IRP address
                                0356   921 ;                       If "startio" event then UCB address
                                0356   922 ;               R5      Address of XWB
                                0356   923 ;               R4      Scratch
                                0356   924 ;               R3      If received message event then scratch
                                0356   925 ;                       If "startio" event then QIO IRP address
                                0356   926 ;               R2      If received msg event then message bytes not yet accounted f
                                0356   927 ;                       If "startio" event then scratch
                                0356   928 ;               R1      If received msg event then ptr to first unprocessed byte in
                                0356   929 ;                       If "startio" event then scratch
                                0356   930 ;               R0      Scratch
                                0356   931 ;
                                0356   932 ;  OUTPUTS:     R0      Status code from the action routine to be returned to
                                0356   933 ;                       the caller of the event dispatcher.
                                0356   934 ;
                                0356   935 ;                 Only R6 and R5 are preserved.
                                0356   936 ;
                                0356   937 ;-
                                0356   938
                                0356   939          ASSUME  XWB$C_NUMSTA EQ 8       ; Assume quadword per event
                                0356   940 NET$EVENT::                             ; Process new event
            59    1E A5    9A   0356   941          MOVZBL  XWB$B_STA(R5),R9        ; Get current state
          54    FDDD CF47   7E   035A   942          MOVAQ   NET$AB_STTAB[R7],R4     ; Get event block address
             59    6449    9A   0360   943          MOVZBL  (R4)[R9],R9             ; Get table entry
     54     59    000000E0 8F   CB   0364   944          BICL3   #NET$M_STAMSK,R9,R4     ; Get action routine index
                                036C   945          ;
                                036C   946          ;
                                036C   947          ;    Dispatch according to the event code.  The action routines
                                036C   948          ;    can assume the following :
                                036C   949          ;
                                036C   950          ;    Inputs:
                                036C   951          ;
                                036C   952          ;       R10     Parameter from caller to action routine
                                036C   953          ;       R9      State information -- not to be touched
                                036C   954          ;       R8      Parameter from caller to action routine
                                036C   955          ;       R7      Event code
                                036C   956          ;       R6      Varies with event
                                036C   957          ;       R5      XWB address
                                036C   958          ;       R4      Scratch
                                036C   959          ;       R3-R1   Varies with event
                                036C   960          ;       R0      Scratch
                                036C   961          ;      (SP)     Return address
                                036C   962          ;
                                036C   963          ;    Returned values:
                                036C   964          ;
```

```
                     036C   965          ;       R8,R7   Garbage
                     036C   966          ;       R6,R5   Preserved
                     036C   967          ;       R4-R1   Garbage
                     036C   968          ;       R0      Status to be returned to caller of dispatcher
                     036C   969          ;
                     036C   970          ;
              00D8   30     036C   971          BSBW    ACT_DISPATCH                ; Call action routine
                     036F   972          ;
                     036F   973  CHANGE_STA:                                      ; Change logical-link state
                 05  EF     036F   974          EXTZV   #NET$C_ACTBITS,-            ;
          54     59  03     0371   975                  #NET$C_STABITS,R9,R4        ; Get next state
        1E A5    54  91     0374   976          CMPB    R4,XWB$B_STA(R5)            ; New state ?
                 29  13     0378   977          BEQL    NET$SCH_MSG                 ; If EQL no, schedule message xmission
                 02  10     037A   978          BSBB    NEW_STATE                   ; Change to new state
                 25  11     037C   979          BRB     NET$SCH_MSG                 ; Schedule message xmission
                     037E   980          ;
                     037E   981          ;
                     037E   982  NEW_STATE:                                       ; Change to new logical-link state
                     037E   983          ;
                     037E   984          ;
                     037E   985          ;   Clear PROGRESS since we are changing states.  The only exceptions
                     037E   986          ;   are if we are coming out of the "closed" state (since PROGRESS has
                     037E   987          ;   been setup to the correct value by the previous action routine) or
                     037E   988          ;   if we are entering the "connect ACK received" state (since we the
                     037E   989          ;   outgoing link timeout period is tied to the receipt of a Connect
                     037E   990          ;   Confirm, not a Connect-ACK).
                     037E   991          ;
                     037E   992          ;
                 00  91     037E   993          CMPB    #XWB$C_STA_CLO,-            ; Coming out of the "closed" state?
               1E A5        0380   994                   XWB$B_STA(R5)              ;
                 08  13     0382   995          BEQL    10$                         ; If EQL, PROGRESS was already init'd
              54     02  91  0384   996          CMPB    #XWB$C_STA_CAR,R4           ; Entering "CAR" state?
                 03  13     0387   997          BEQL    10$                         ; If EQL yes, do not re-init PROGRESS
             -52 A5  B4     0389   998          CLRW    XWB$W_PROGRESS(R5)          ; Init progress count
        1E A5    54  90     038C   999  10$:    MOVB    R4,XWB$B_STA(R5)            ; Change state
                     AA     0390  1000          BICW    #XWB$M_STS_TID!-            ; Always clear "timer id valid" and
                            0391  1001                   XWB$M_STS_DIS,-            ; "disconnect pending" flags
        0E A5    09         0391  1002                   XWB$W_STS(R5)              ;
                            0394  1003                                              ;
1C A5    FD93 CF44     AA     0394  1004          BICW    NET$AW_FLG_CLRM[R4],XWB$W_FLG(R5)  ; Clear indicated flags
1C A5    FD7C CF44     A8     039B  1005          BISW    NET$AW_FLG_SETM[R4],XWB$W_FLG(R5)  ; Set indicated flags
                     05     03A2  1006          RSB                                        ; Done
                            03A3  1007
```

```
                        03A3  1009 .SBTTL  NET$SCH_MSG     - schedule message transmission
                        03A3  1010 ;+
                        03A3  1011 ;
                        03A3  1012 ;  The following flags are used to cause control messages to be setup when the
                        03A3  1013 ;  control message cell in the XWB becomes available.  As each message is
                        03A3  1014 ;  entered into this contol message cell, the corresponding bit is cleared.
                        03A3  1015 ;
                        03A3  1016 ;  These flags are listed in the order of their priority.
                        03A3  1017 ;
                        03A3  1018 ;
                        03A3  1019 ;       XWB$V_FLG_TBPR - Set whenever the receive back pressure state needs to
                        03A3  1020 ;                        be toggled.
                        03A3  1021 ;
                        03A3  1022 ;       XWB$V_FLG_IAVL - Set whenever a new xmit interrupt IRP makes it to the
                        03A3  1023 ;                        head of the LSB queue and the partner's flow control
                        03A3  1024 ;                        on the INT/LS subchannel will let us send the message.
                        03A3  1025 ;
                        03A3  1026 ;       XWB$V_FLG_SIFL - Set whenever an INTERRUPT message has been sent to the
                        03A3  1027 ;                        user's mailbox.
                        03A3  1028 ;
                        03A3  1029 ;       XWB$V_FLG_SDFL - Set whenever the inactivity timer fires in order to
                        03A3  1030 ;                        maintain a minimal amount of traffic on the link to
                        03A3  1031 ;                        see if the remote node is still active.
                        03A3  1032 ;
                        03A3  1033 ;
                        03A3  1034 ;  Whether or not a new Link-service/Interrupt message is setup in the XWB
                        03A3  1035 ;  cell, XWB$W_FLG(R5) is scanned to see if any work needs to be done.  If
                        03A3  1036 ;  so, and if the XWB fork block is not in use, control is passed to
                        03A3  1037 ;  NET$SOLICIT.
                        03A3  1038 ;
                        03A3  1039 ;
                        03A3  1040 ;  INPUTS:      R5      XWB address
                        03A3  1041 ;                R4-R0   Scratch
                        03A3  1042 ;
                        03A3  1043 ;  OUTPUTS:     R4-R0   Garbage
                        03A3  1044 ;
                        03A3  1045 ;                All other registers are preserved.
                        03A3  1046 ;
                        03A3  1047 ;-
                        03A3  1048 NET$SCH_MSG::                                    ; Schedule message transmission
                        03A3  1049
                        03A3  1050          ASSUME  XWB$V_FLG_IAVL  EQ  1+XWB$V_FLG_TBPR
                        03A3  1051          ASSUME  XWB$V_FLG_SIFL  EQ  1+XWB$V_FLG_IAVL
                        03A3  1052          ASSUME  XWB$V_FLG_SDFL  EQ  1+XWB$V_FLG_SIFL
                        03A3  1053
50   1C A5    04   0B   EA 03A3  1054          FFS     #XWB$V_FLG_TBPR,#4,XWB$W_FLG(R5),R0 ; Find message to build
                        74   13 03A9  1055          BEQL    90$                         ; If EQL then none
      6F 6C A5    04   E0 03AB  1056          BBS     #NSP$V_FLW_INUSE,XWB$B_X_FLW(R5),90$; If BS, msg cell is in use
         6C A5    10   90 03B0  1057          MOVB    #NSP$M_FLW_INUSE,XWB$B_X_FLW(R5)    ; Claim the cell, clear flags
               6D A5    94 03B4  1058          CLRB    XWB$B_X_FLWCNT(R5)                  ; Init flow request count
                        03B7  1059                                                      ;
      52    00D4 C5    9E 03B7  1060          MOVAB   XWB$T_LI(R5),R2                     ; Setup LSB pointer
                        62   B6 03BC  1061          INCW    LSB$W_LUX(R2)               ; Get next sequence number
      62    F000 8F    AA 03BE  1062          BICW    #^X<F000>,LSB$W_LUX(R2)             ; Mask off junk bits
         04 A2    62   B0 03C3  1063          MOVW    LSB$W_LUX(R2),LSB$W_HXS(R2)         ; It's sendable now
      00 1C A5    50   E5 03C7  1064          BBCC    R0,XWB$W_FLG(R5),20$                ; Clear the work bit
                        03CC  1065                                                      ;
```

```
                     03CC  1066 20$:    $DISPATCH  R0,-                          ; Dispatch on work bit
                     03CC  1067         <-
                     03CC  1068           <XWB$V_FLG_IAVL,   50$>,-              ; INTerrupt msg
                     03CC  1069           <XWB$V_FLG_SIFL,   40$>,-              ; INTerrupt flow control msg
                     03CC  1070           <XWB$V_FLG_SDFL,   80$>,-              ; DATA flow control msg
                     03CC  1071         >                                        ; Fall thru if
                     03D6  1072                                                  ;   R0=XWB$V_FLG_TBPR
                     03D6  1073         ;
                     03D6  1074         ;
                     03D6  1075         ;    Setup for DATA channel control message to toggle the local
                     03D6  1076         ;    receiver back pressure state
                     03D6  1077         ;
                     03D6  1078         ;
           50   01   90  03D6  1079         MOVB   #NSP$M_FLW_XOFF,R0                ; Setup for "stop flow" message
     13 0E A5   06   E3  03D9  1080         BBCS   #XWB$V_STS_RBP,XWB$W_STS(R5),30$  ; If BC, not back-pressured off
  0E A5   0040 8F   AA  03DE  1081         BICW   #XWB$M_STS_RBP,XWB$W_STS(R5)      ; Mark receiver as having its
                     03E4  1082                                                  ; back-pressured relaxed
           50   02   90  03E4  1083         MOVB   #NSP$M_FLW_XON,R0                 ; Setup for "start flow" msg
                     03E7  1084         ;
                     03E7  1085         ;
                     03E7  1086         ;    Force a NAK on the DATA sub-channel in order to reset its sequence
                     03E7  1087         ;    numbers.  Ordinarily, NAK's are undesireable in a routing
                     03E7  1088         ;    environment since they could contribute to congestion problems.
                     03E7  1089         ;    But that is generally true for NAK's sent due to receiving messages
                     03E7  1090         ;    out of order (a message received out of order is often due to
                     03E7  1091         ;    network congestion loss of an earlier packet).
                     03E7  1092         ;
                     03E7  1093         ;    The NAK is not absolutely necessary, but failure to send it will
                     03E7  1094         ;    mean an inordinate delay since the remote sequence numbers will
                     03E7  1095         ;    not be reset when back-pressure is subsequently relaxed (only a
                     03E7  1096         ;    NAK or timeout resets the sequence numbers).
                     03E7  1097         ;
                     03E7  1098         ;       NOTE:   Since XWB$M_FLG_SDACK is less than XMB$M_FLG_SLI, the
                     03E7  1099         ;               NAK will be sent just before the back-pressure message.
                     03E7  1100         ;               This is the desired order.
                     03E7  1101         ;
                     03E7  1102         ;
                     03E7  1103         ;
     1C A5   08   A8  03E7  1104         BISW   #XWB$M_FLG_SDACK,XWB$W_FLG(R5)    ; Force NAK on the DATA channel
  0E A5   0100 8F   A8  03EB  1105         BISW   #XWB$M_STS_DTNAK,XWB$W_STS(R5)    ;   in order to reset it
     6C A5   50   88  03F1  1106 30$:     BISB    R0,XWB$B_X_FLW(R5)               ; Set remaining control flags
  1B 1C A5   0D   E5  03F5  1107         BBCC   #XWB$V_FLG_SIFL,XWB$W_FLG(R5),80$; Piggy-back INT flow control
                     03FA  1108                                                  ; message if possible
                     03FA  1109 40$:    ;
                     03FA  1110         ;
                     03FA  1111         ;    Setup Interrupt flow-control message.
                     03FA  1112         ;
                     03FA  1113         ;
     6C A5   04   88  03FA  1114         BISB   #NSP$M_FLW_LISUB,XWB$B_X_FLW(R5) ; Flow control for LI channel
        6D A5   96   03FE  1115         INCB    XWB$B_X_FLWCNT(R5)               ; Ask for one more INT message
     00FD C5   96   0401  1116         INCB    XWB$T_LI+LSB$B_R_CXBQUO(R5)      ; And allow it to be received
           0E   11   0405  1117         BRB     80$                              ; Schedule msg for transmission
                     0407  1118 50$:    ;
                     0407  1119         ;
                     0407  1120         ;    Setup for Interrupt message
                     0407  1121         ;
                     0407  1122         ;
```

```
                                0415  1123          BISB    #NSP$M_FLW_INT,XWB$B_X_FLW(R5)   ; Not "link service" message
                             50   10 A2  D0  040B  1124          MOVL     LSB$L_X_PND(R2),R0              ; Get IRP
                          20 A0   0040 8F  AA  040F  1125          BICW    #IO$M_INTERRUPT,IRP$W_FUNC(R0)   ; Indicate state change
                                0415  1126 80$:     ;
                                0415  1127          ;
                                0415  1128          ;    Schedule the message for transmission.
                                0415  1129          ;
                                0415  1130          ;
                             1C A5   10  A8  0415  1131          BISW    #XWB$M_FLG_SLI,XWB$W_FLG(R5)     ; We've got a message to send
                          1C A5   4000 8F  AA  0419  1132          BICW    #XWB$M_FLG_SDFL,XWB$W_FLG(R5)    ; Whatever has just been built
                                041F  1133                                                           ; satisfies the need to send the
                                041F  1134                                                           ; backround inactivity message
                                041F  1135 90$:     ;
                                041F  1136          ;
                                041F  1137          ;    Unless there are any wait conditions pending, solicit permission
                                041F  1138          ;    from the Routing layer to transmit a message.
                                041F  1139          ;
                                041F  1140          ;
                             0A   00  EA  041F  1141          FFS     #0,#XWB$V_FLG_CLO+1,-             ; Get work bit
                          50   1C A5      0422  1142                      XWB$W_FLG(R5),R0             ;
                       0E FDBA CF   50  E1  0425  1143          BBC      R0,NET$GL_WORKBITS,200$         ; Br if no work to be done
                         08 0E A5   02  E2  042B  1144          BBSS    #XWB$V_STS_SOL,XWB$W_STS(R5),100$; If BS, fork block in use
                                0430  1145                                                           ;
                                   55   DD  0430  1146          PUSHL    R5                              ; Save XWB address
                                FBCB'   30  0432  1147          BSBW     NSP$SOLICIT                     ; Get permission to transmit
                                   55 8ED0  0435  1148          POPL     R5                              ; Restore XWB address
                                      05  0438  1149 100$:     RSB
                                0439  1150
                          FA 0E A5   03   E1  0439  1151 200$:     BBC     #XWB$V_STS_DIS,XWB$W_STS(R5),100$; If BC, disconnect not pending
                                 0381   30  043E  1152          BSBW     NET$CHK_X_IDLE                  ; Is XWB ready for disconnect ?
                                F4 50   E9  0441  1153          BLBC     R0,100$                         ; If LBC then no
                                  FEA5  31  0444  1154          BRW      NET$FORK                        ; Attempt to resume disconnect
                                0447  1155
```

```
                     0447  1157
                     0447  1158 ACT_DISPATCH:                            ; Dispatch action routine
                     0447  1159
                     0447  1160          $DISPATCH       TYPE=B,R4,              -; R4 contains the action index
                     0447  1161          <-
                     0447  1162            <ACT$_ABORT,        ACT$ABORT>,       -;
                     0447  1163            <ACT$_BUG,          ACT$BUG>,         -;
                     0447  1164            <ACT$_CANLNK,       ACT$CANLNK>,      -;
                     0447  1165            <ACT$_CONFIRM,      ACT$CONFIRM>,     -;
                     0447  1166            <ACT$_DEACCESS,     ACT$DEACCESS>,    -;
                     0447  1167            <ACT$_ENT_RUN,      ACT$ENT_RUN>,     -;
                     0447  1168            <ACT$_INITIATE,     ACT$INITIATE>,    -;
                     0447  1169            <ACT$_LOG,          ACT$LOG>,         -;
                     0447  1170            <ACT$_NOP,          ACT$NOP>,         -;
                     0447  1171            <ACT$_RES_DISC,     ACT$RES_DISC>,    -;
                     0447  1172            <ACT$_RCV_CA,       ACT$RCV_CA>,      -;
                     0447  1173            <ACT$_RCV_CC,       ACT$RCV_CC>,      -;
                     0447  1174            <ACT$_RCV_CI,       ACT$RCV_CI>,      -;
                     0447  1175            <ACT$_RCV_CR,       ACT$RCV_CR>,      -;
                     0447  1176            <ACT$_RCV_DATA,     ACT$RCV_DATA>,    -;
                     0447  1177            <ACT$_RCV_DTACK,    ACT$RCV_DTACK>,   -;
                     0447  1178            <ACT$_RCV_DX        ACT$RCV_DX>,      -;
                     0447  1179            <ACT$_RCV_LI,       ACT$RCV_LI>,      -;
                     0447  1180            <ACT$_RCV_LIACK,    ACT$RCV_LIACK>,   -;
                     0447  1181            <ACT$_RCV_RTS,      ACT$RCV_RTS>,     -;
                     0447  1182            <ACT$_RTS_NLT,      ACT$RTS_NLT>,     -;
                     0447  1183            <ACT$_SHRLNK,       ACT$SHRLNK>,      -;
                     0447  1184            <ACT$_SSABORT,      ACT$SSABORT>,     -;
                     0447  1185          >
01   11              0477  1186          BRB     ACT$BUG                         ; If unknown, bug
                     0479  1187
```

```
                     0479  1189 .SBTTL  ACT$NOP          - Null action routine
                     0479  1190 .SBTTL  ACT$BUG          - BUG_CHECK action routine
                     0479  1191 .SBTTL  ACT$LOG          - Log-event action routine
                     0479  1192 .SBTTL  ACT$NOLINK       - Report "SS$_FILNOTACC"
                     0479  1193 .SBTTL  ACT$SSABORT      - Abort QIO since link was disconnected
                     0479  1194
                05   0479  1195 ACT$NOP:          RSB
                     047A  1196 ACT$BUG:          BUG_CHECK NETNOSTATE,FATAL
                01   047E  1197 ACT$LOG:          NOP
                01   047F  1198                   NOP
                     0480  1199
                01   0480  1200                   NOP
                01   0481  1201                   NOP
                01   0482  1202                   NOP
                05   0483  1203                   RSB
                     0484  1204
                     0484  1205
      38 A3  00AC 8F 3C   0484  1206 ACT$NOLINK:     MOVZWL  #SS$_FILNOTACC,IRP$L_IOST1(R3)
                05   048A  1207                   RSB
                     048B  1208
                     048B  1209 ACT$SHRLNK:       ;&nyi
         38 A3  2C 3C   048B  1210 ACT$SSABORT:     MOVZWL  #SS$_ABORT,IRP$L_IOST1(R3)
                05   048F  1211                   RSB
                     0490  1212
                     0490  1213
```

```
                        0490  1215 .SBTTL  NET$STARTIO     - Start I/O operation
                        0490  1216 ;+
                        0490  1217 ;
                        0490  1218 ;  This routine is entered when the associated unit is idle and a packet
                        0490  1219 ;  is available for processing.  The IRP$L_WIND field is used to locate the
                        0490  1220 ;  associated XWB.
                        0490  1221 ;
                        0490  1222 ;
                        0490  1223 ;  INPUTS:       R5      UCB address
                        0490  1224 ;                R4      PCB address
                        0490  1225 ;                R3      IRP address
                        0490  1226 ;
                        0490  1227 ;  OUTPUTS:       *** TBS ***
                        0490  1228 ;
                        0490  1229 ;-
                        0490  1230 NET$STARTIO:                                         ; Process next IRP
            OFE0 8F   BB 0490  1231          PUSHR   #^M<R5,R6,R7,R8,R9,R10,R11>     ; Setup "event" context
                44    10 0494  1232          BSBB    PROC_IO                         ; Process the I/O function
              05 55   E8 0496  1233          BLBS    R5,20$                          ; If LBS, no event to process
                 5B   D4 0499  1234          CLRL    R11                             ; Say "can't go to IPL 2"
               FEB8   30 049B  1235          BSBW    NET$EVENT                       ; Process event in R7
            OFE0 8F   BA 049E  1236 20$:     POPR    #^M<R5,R6,R7,R8,R9,R10,R11>     ; Return to UCB 'fork' context
                      04A2  1237                                                    ;
        53    58 A5   D0 04A2  1238          MOVL    UCB$L_IRP(R5),R3                ; Get IRP
                1F    13 04A6  1239          BEQL    50$                             ; If EQL then its been queued
                      04A8  1240                                                    ; or suspended, start next I/O
                      04A8  1241          ;
                      04A8  1242          ;
                      04A8  1243          ;    Deallocate misc. buffer
                      04A8  1244          ;
                      04A8  1245          ;
     0C 2A A3    03   E1 04A8  1246          BBC     #IRP$V_COMPLX,IRP$W_STS(R3),40$ ; If BC, IRP$L_DIAGBUF does not
                      04AD  1247                                                    ; point to a NETDRIVER buffer
        50    4C A3   D0 04AD  1248          MOVL    IRP$L_DIAGBUF(R3),R0            ; Get buffer
                06    18 04B1  1249          BGEQ    40$                             ; If GEQ then none
              4C A3   D4 04B3  1250          CLRL    IRP$L_DIAGBUF(R3)               ; Detach it
               0895   30 04B6  1251          BSBW    NET$DEALLOCATE                  ; Deallocate block in R0
                      04B9  1252 40$:     ;
                      04B9  1253          ;
                      04B9  1254          ;    Start next I/O.
                      04B9  1255          ;
                      04B9  1256          ;
        50    38 A3   D0 04B9  1257          MOVL    IRP$L_IOST1(R3),R0              ; First  IOSB longword
        51    44 A5   D0 04BD  1258          MOVL    UCB$L_DEVDEPEND(R5),R1          ; Second IOSB longword
                      04C1  1259          REQCOM                                  ; Complete I/O, start next IRP
        53    4C B5   0F 04C7  1260 50$:     REMQUE  @UCB$L_IOQFL(R5),R3             ; Get next IRP
                06    1D 04CB  1261          BVS     60$                             ; If VS then none
    00000000'GF       17 04CD  1262          JMP     G^IOC$INITIATE                  ; Deliver IRP to driver
  64 A5    0100 8F    AA 04D3  1263 60$:     BICW    #UCB$M_BSY,UCB$W_STS(R5)        ; Free up the UCB
                05    05 04D9  1264          RSB                                     ; Return to Exec
                      04DA  1265
                      04DA  1266 PROC_IO::
                      04DA  1267          ;
                      04DA  1268          ;    Move the UCB to R6 and the XWB (if any) to R5 and dispatch
                      04DA  1269          ;    on function code with:
                      04DA  1270          ;
                      04DA  1271          ;       R10-R7  Scratch
```

```
                                                E 14
                                04DA  1272          ;       R6      UCB address
                                04DA  1273          ;       R5      XWB address if LBC, else garbage
                                04DA  1274          ;       R3      IRP address
                                04DA  1275          ;       R2-R0   Scratch
                                04DA  1276          ;
                                04DA  1277          ;
                    56    55    D0  04DA  1278          MOVL    R5,R6                                   ; Copy UCB to safe register
                 55    18 A3    D0  04DD  1279          MOVL    IRP$L_WIND(R3),R5                       ; Get XWB, if any
                          03    19  04E1  1280          BLSS    10$                                     ; If LSS, XWB is in system space
                    55    01    C8  04E3  1281          BISL    #1,R5                                   ; Else, invalidate window ptr
              38 A3    00F4  8F    3C  04E6  1282  10$:   MOVZWL  #SS$_ILLIOFUNC,IRP$L_IOST1(R3)       ; Assume fct not supported
        57    20 A3    FFFFFFC0 8F   CB  04EC  1283          BICL3   #^C<IO$M_FCODE>,IRP$W_FUNC(R3),R7 ; Get code without modifiers
                                04F5  1284
                                04F5  1285          $DISPATCH  R7,TYPE=B,-                          ; Process I/O
                                04F5  1286          <-
                                04F5  1287            <IO$_ACCESS,     NET$ACCESS>,-                ; Connect Requests
                                04F5  1288            <IO$_DEACCESS,   NET$DEACCESS>,-              ; Disconnect Requests
                                04F5  1289            <IO$_ACPCONTROL, NET$CONTROL>,-               ; ACP Control function
                                04F5  1290          >                                               ; Else, fall thru
                    55    01    D0  0507  1291          MOVL    #1,R5                                   ; Set low bit to prevent event
                                05  050A  1292          RSB                                             ; dispatching and return
                                    050B  1293
```

```
                                050B  1295 .SBTTL  NET$FDT_SETMODE - Process IO$_SETMODE request
                                050B  1296 ;+
                                050B  1297 ;
                                050B  1298 ; *** tbs ***
                                050B  1299 ;
                                050B  1300 ;-
                                050B  1301 NET$FDT_SETMODE:                                     ; Process IO$_SETMODE function
          51    6C    D0      050B  1302          MOVL    P1(AP),R1                             ; Get characteristics buffer
                0B    13      050E  1303          BEQL    10$                                   ; If EQL then none
                              0510  1304          IFNORD  #8,(R1),50$                           ; Br on access violation
    44 A5     04 A1    D0      0516  1305          MOVL    4(R1),UCB$L_DEVDEPEND(R5)             ; Set mailbox mask
50  18 A3     01    CB      051B  1306 10$:       BICL3   #1,IRP$L_WIND(R3),R0                  ; Get XWB address
                00    18      0520  1307          BGEQ    40$                                   ; If GEQ then none
                              0522  1308          ;
                              0522  1309          ;
                              0522  1310          ;    This was used for "maintenance" mode.  Now available for
                              0522  1311          ;    future functions.
                              0522  1312          ;
                              0522  1313          ;
          51    44 A5     D0      0522  1314 40$:       MOVL    UCB$L_DEVDEPEND(R5),R1          ; Get device dependent info
              50    01    3C      0526  1315          MOVZWL  S^#SS$_NORMAL,R0                  ; Setup I/O status
       00000000'GF    17      0529  1316          JMP     G^EXE$FINISHIO                        ; Return success
                              052F  1317
              50    0C    3C      052F  1318 50$:       MOVZWL  #SS$_ACCVIO,R0                  ; Setup I/O status
       00000000'GF    17      0532  1319          JMP     G^EXE$ABORTIO                         ; Abort I/O
                              0538  1320
```

```
                     0538  1322 .SBTTL  NET$FDT_CONTROL - IO$_ACPCONTROL FDT processing
                     0538  1323 .SBTTL  NET$CONTROL     - IO$_ACPCONTROL "startio" processing
                     0538  1324 ;+
                     0538  1325 ;
                     0538  1326 ;  The FDT routine simply routes the IRP through the Exec to the ACP.  The Exec
                     0538  1327 ;  builds a "complex buffer" describing the control function.  The ACP will
                     0538  1328 ;  requeue any IRP to the driver if it does not recognize the control function.
                     0538  1329 ;  The driver has been designed to handle some of its own control functions
                     0538  1330 ;  since many are protocol or control block format specific.
                     0538  1331 ;
                     0538  1332 ;
                     0538  1333 ;  INPUTS:      R5      UCB Address
                     0538  1334 ;               R4      PCB Address
                     0538  1335 ;               R3      IRP Address
                     0538  1336 ;
                     0538  1337 ;  OUTPUTS:     R5      Unchanged
                     0538  1338 ;               R0      I/O status
                     0538  1339 ;
                     0538  1340 ;-
                     0538  1341 NET$FDT_CONTROL:                                    ; FDT phase for IO$_ACPCONTROL
           4C A3  D4 0538  1342         CLRL    IRP$L_DIAGBUF(R3)                 ; Zero misc buffer pointer
        18 A3  01  CA 053B  1343         BICL    #1,IRP$L_WIND(R3)                 ; Always clear interlock flag
                     053F  1344         ASSUME  PHD$Q_PRIVMSK  EQ  0              ;
      40 A3  6C B4  7D 053F  1345         MOVQ    @PCB$[_PHD(R4),IRP$Q_NT_PRVMSK(R3); Store privilege mask
        00000000'GF  17 0544  1346         JMP     G^ACP$MODIFY                      ; Continue in EXEC
                     054A  1347
                     054A  1348
                     054A  1349 ;
                     054A  1350 ;  INPUTS:      *** tbs ***
                     054A  1351 ;  OUTPUTS:     *** tbs ***
                     054A  1352 ;
                     054A  1353 ;
                     054A  1354 NET$CONTROL:                                    ; "Startio" for IO$_ACPCONTROL
  08 2A A3  03  E1   054A  1355         BBC     #IRP$V_COMPLX,IRP$W_STS(R3),10$ ; If BC, part of $CANCEL
     50  2C B3  D0   054F  1356         MOVL    @IRP$L_SVAPTE(R3),R0              ; Get ptr to window descriptor
         60  D4      0553  1357         CLRL    (R0)                              ; Don't affect window upon
         1F  11      0555  1358         BRB     50$                               ;  I/O completion
                     0557  1359 10$:     ;
                     0557  1360         ;
                     0557  1361         ;   The user is getting ready to issue an IO$_DEACCESS QIO to break the
                     0557  1362         ;   link.  In order for the IO$_DEACCESS to be sent to the driver, the
                     0557  1363         ;   channel's outstanding I/O count (CCB$W_IOC) must be zero.  Hence the
                     0557  1364         ;   receiver must be run-down and any outstanding receive IRP's aborted.
                     0557  1365         ;
                     0557  1366         ;
  38 A3  01  3C      0557  1367         MOVZWL  #SS$_NORMAL,IRP$L_IOST1(R3)       ; Set I/O status
     55  01  CA      055B  1368         BICL    #1,R5                             ; Clear interlock bit
         2F  19      055E  1369         BLSS    70$                               ; If LSS then valid XWB
                     0560  1370         ;
                     0560  1371         ;
                     0560  1372         ;    Scan LTB to find XWB with an access pending for this channel
                     0560  1373         ;
                     0560  1374         ;
     50  34 A6  D0  0560  1375         MOVL    UCB$L_VCB(R6),R0                  ; Get RCB
         10  13      0564  1376         BEQL    50$                               ; If EQL then none
     50  24 A0  D0  0566  1377         MOVL    RCB$L_PTR_LTB(R0),R0              ; Get LTB
         0A  13      056A  1378         BEQL    50$                               ; If EQL then none
```

```
                55   E0 A0    9E   056C  1379                MOVAB  -XWB$L_LINK+LTB$L_XWB(R0),R5     ; Prepare for XWB scan
                55   2C A5    D0   0570  1380  20$:           MOVL    XWB$L_LINK(R5),R5               ; Get next XWB
                          04    12   0574  1381                BNEQ    60$                             ; If EQL then none left
                     55   01    88   0576  1382  50$:           BISB    #1,R5                           ; Prevent event dispatching
                                05   0579  1383                RSB                                     ; Done
                                     057A  1384
              50    0080 C5    D0   057A  1385  60$:           MOVL    XWB$L_IRP_ACC(R5),R0            ; Get suspended IRP, if any
                                EF   13   057F  1386           BEQL    20$                             ; If EQL none, loop
           0C A3    0C A0    D1   0581  1387                CMPL    IRP$L_PID(R0),IRP$L_PID(R3)     ; Belong to this process ?
                                E8   12   0586  1388           BNEQ    20$                             ; Loop if NEQ
           28 A3    28 A0    B1   0588  1389                CMPW    IRP$W_CHAN(R0),IRP$W_CHAN(R3)   ; Same channel ?
                                E1   12   058D  1390           BNEQ    20$                             ; If NEQ, loop
                                     058F  1391  70$:           ;
                                     058F  1392                ;
                                     058F  1393                ;   The transmitter is not automatically run-down since the user may be
                                     058F  1394                ;   preparing a "synchronous" disconnect -- i.e., disconnect after the
                                     058F  1395                ;   final data segment has been ACK'd.  The manner in which pipelining
                                     058F  1396                ;   has been implemented allows user transmit IRP's to be sent to I/O
                                     058F  1397                ;   completion before the corresponding CXB's have been ACK'd  (or even
                                     058F  1398                ;   sent).  Therefore, the user might issue a call to $CANCEL mistakenly
                                     058F  1399                ;   thinking that the final message has be ACK'd.  Hence $CANCEL should
                                     058F  1400                ;   allow the transmit CXB's to be ACK'd in their normal fashion.
                                     058F  1401                ;
                                     058F  1402                ;   Therefore, drain the receiver of all IRP's and CXB's.  If there
                                     058F  1403                ;   are any transmit IRP's on the queue, then the disconnect is not
                                     058F  1404                ;   synchronous, and the transmitter queue must be drained as well.
                                     058F  1405                ;
                                     058F  1406                ;
        0B 0E A5      04    E0   058F  1407                BBS     #XWB$V_STS_CON,XWB$W_STS(R5),80$ ; If BS, IO$_ACCESS pending
                    038A    30   0594  1408                BSBW     DRAIN_RCV                      ; Drain the receiver
                      50    D4   0597  1409                CLRL     R0                             ; Init R0 for CHK_X_IRP call
                    0235    30   0599  1410                BSBW     CHK_X_IRP                      ; Any Xmt IRP's
                   D7 50    E8   059C  1411                BLBS     R0,50$                         ; If LBS, no
                57    0D    3C   059F  1412  80$:           MOVZWL   #NETEVT$_CANLNK,R7             ; Force link to break
                                05   05A2  1413                RSB                                     ; Done
                                     05A3  1414
                                     05A3  1415
```

```
                     05A3  1417 .SBTTL  NET$FDT_ACCESS  - IO$_ACCESS  FDT processing
                     05A3  1418 .SBTTL  NET$ACCESS      - IO$_ACCESS "startio" processing
                     05A3  1419 ;++
                     05A3  1420 ;
                     05A3  1421 ;  NET$FDT_ACCESS passes the IRP through the EXEC, where the user parameters
                     05A3  1422 ;  are packaged into a "complex buffer", to the ACP.  The ACP reads the user
                     05A3  1423 ;  connect info to build an Internal Connect Block (ICB) which it attaches to
                     05A3  1424 ;  the IRP$L_DIAGBUF field of the IRP and requeues the IRP to the driver.  The
                     05A3  1425 ;  role of the ACP is to lookup default access control (username, password,
                     05A3  1426 ;  account) information in its data base and to translate node and object names
                     05A3  1427 ;  to numbers.
                     05A3  1428 ;
                     05A3  1429 ;  NET$ACCESS reads the ICB and determines the type of connect.  It builds an
                     05A3  1430 ;  XWB for connect initiate events and locates an already existing XWB for all
                     05A3  1431 ;  others.  NET$ACCESS stores the appropriate event code in R7 and returns
                     05A3  1432 ;  expecting the caller to call the event dispatcher.
                     05A3  1433 ;
                     05A3  1434 ;  It should be noted that the size of the XWB is not charged against the user
                     05A3  1435 ;  byte count or byte limit quotas.  It is assumed that these quotas are at
                     05A3  1436 ;  least partly used to limit a run away process and that the file quota of a
                     05A3  1437 ;  process, against which logical-links are charged, is a sufficient mechanism.
                     05A3  1438 ;
                     05A3  1439 ;
                     05A3  1440 ;
                     05A3  1441 ;  INPUTS:       *** tbs ***
                     05A3  1442 ;
                     05A3  1443 ;  OUTPUT:       *** tbs ***
                     05A3  1444 ;
                     05A3  1445 ;
                     05A3  1446 ;-
                     05A3  1447 NET$FDT_ACCESS:                                      ; IO$_ACCESS  "FDT" processing
                     05A3  1448
                     05A3  1449          ASSUME  PHD$Q_PRIVMSK EQ 0
                     05A3  1450
40 A3    6C B4   7D  05A3  1451          MOVQ    @PCB$L_PHD(R4),IRP$Q_NT_PRVMSK(R3) ; Store priv mask in IRP
         4C A3   D4  05A8  1452          CLRL    IRP$L_DIAGBUF(R3)                   ; Indicate no ICB
  00000000'GF    17  05AB  1453          JMP     G^ACP$ACCESSNET                     ; Continue in EXEC
                     05B1  1454
                     05B1  1455
                     05B1  1456 NET$ACCESS:                                          ; IO$_ACCESS "startio" processing
       023A      30  05B1  1457          BSBW    GET_WNDSC                           ; Get CCB$L_WIND image descr.
         67      D4  05B4  1458          CLRL    (R7)                                ; Init CCB$L_WIND image
    2A A3        02  05B6  1459          BISW    #IRP$M_FUNC,IRP$W_STS(R3)           ; Mark for write back
    32 A3        01  05BA  1460          MOVZWL  #1,IRP$L_BCNT(R3)                   ; Write back one descriptor
       55        56  05BE  1461          MOVL    R6,R5                               ; Copy UCB addr for subroutines
       58        53  05C1  1462          MOVL    R3,R8                               ; Copy IRP address to safe reg
    54    4C A8   D0  05C4  1463          MOVL    IRP$L_DIAGBUF(R8),R4                ; Get ICB pointer
         3C        18  05C8  1464          BGEQ    80$                                 ; If GEQ, its an error code
                     05CA  1465 10$:      ;
                     05CA  1466           ;
                     05CA  1467           ;      IO$_ACCESS made it successfully through the ACP
                     05CA  1468           ;
                     05CA  1469           ;
    53    02 A4   3C  05CA  1470          MOVZWL  ICB$W_LOCLNK(R4),R3                 ; Get local link address
         06C6     30  05CE  1471          BSBW    XWB_LOCLNK                          ; Find associated XWB
       2D 55      E8  05D1  1472          BLBS    R5,60$                              ; Br if XWB was not found
0C A8    34 A5   D1  05D4  1473          CMPL    XWB$L_PID(R5),IRP$L_PID(R8)         ; PIDs match ?
```

```
                1F   12   05D9  1474          BNEQ    55$                             ; Br if they don't
             3C A5   B5   05DB  1475          TSTW    XWB$W_REMLNK(R5)                ; Does remote link id exit ?
                0B   12   05DE  1476          BNEQ    30$                             ; Connect Confirm if NEQ
                          05E0  1477          ;
                          05E0  1478          ;
                          05E0  1479          ;    Connect Initiate
                          05E0  1480          ;
                          05E0  1481          ;
          04 A4    01     A1   05E0  1482          ADDW3   #1,ICB$W_TIM_OCON(R4),-         ; Setup outbound connect timer
                50 A5                05E4  1483                  XWB$W_TIMER(R5)         ; (+1 for possible clock skew)
                57   0F   9A   05E6  1484          MOVZBL  #NETEVT$_CIA,R7                 ; Set 'connect initiate access'
                0B   11   05E9  1485          BRB     50$                             ; Finish in common
                          05EB  1486  30$:      ;
                          05EB  1487          ;
                          05EB  1488          ;    Connect Confirm
                          05EB  1489          ;
                          05EB  1490          ;
                57   10   9A   05EB  1491  40$:    MOVZBL  #NETEVT$_CCA,R7                 ; Set 'connect confirm access'
          03 20 A8   08   E1   05EE  1492          BBC     #IO$V_ABORT,IRP$W_FUNC(R8),50$  ; If BC, not Connect Reject
                57   11   9A   05F3  1493          MOVZBL  #NETEVT$_CRA,R7                 ; Set 'connect reject access'
                          05F6  1494  50$:      ;
                          05F6  1495          ;
                          05F6  1496          ;    Because the low bit of R5 is clear, the XWB will considered to
                          05F6  1497          ;    be valid and the event in R7 will be processed.
                          05F6  1498          ;
                          05F6  1499          ;
                53   58   D0   05F6  1500          MOVL    R8,R3                           ; Setup IRP address
                     05   05F9  1501          RSB                                     ; Return with LBC in R5
                          05FA  1502
                          05FA  1503  ;
                          05FA  1504  ;  Unsuccessful access
                          05FA  1505  ;
          54    0840 8F    3C   05FA  1506  55$:    MOVZWL  #SS$_DEVALLOC,R4                ; Setup error code
                     05   11   05FF  1507          BRB     80$                             ; Continue
          54    20DC 8F    3C   0601  1508  60$:    MOVZWL  #SS$_CONNECFAIL,R4              ; Setup error code
                53   58   D0   0606  1509  80$:    MOVL    R8,R3                           ; Setup IRP pointer
             38 A3   54   D0   0609  1510          MOVL    R4,IRP$L_IOST1(R3)              ; Store error code
                55   01   D0   060D  1511          MOVL    #1,R5                           ; Tell CLEANUP_ACCESS ''no XWB''
                     0201 30   0610  1512          BSBW    CLEANUP_ACCESS                  ; Restore quota
                          05   0613  1513          RSB                                     ; On return goto REQCOM
                          0614  1514
                          0614  1515  100$:    BUG_CHECK  NETNOSTATE,FATAL
                          0618  1516
```

```
                     0618  1518 .SBTTL  ACT$INITIATE    - Connect Initiate action routine
                     0618  1519 .SBTTL  ACT$CONFRIM     - Connect Confirm action routine
                     0618  1520 ;+
                     0618  1521 ;
                     0618  1522 ;  These action routines resume processing the event setup by NET$ACCESS.
                     0618  1523 ;  ACT$INITIATE assumes that a Connect Initiate message will be built
                     0618  1524 ;  and sent.  ACT$CONFIRM is used when a received connect is being either
                     0618  1525 ;  accepted or rejected and assumes that eihter a Connect Confirm or a
                     0618  1526 ;  Disconnect Initiate message will be built and sent.
                     0618  1527 ;
                     0618  1528 ;
                     0618  1529 ;  INPUTS:       R8      Scratch
                     0618  1530 ;                R7      Event code
                     0618  1531 ;                R6      UCB address
                     0618  1532 ;                R5      XWB address
                     0618  1533 ;                R4      Scratch
                     0618  1534 ;                R3      IRP address
                     0618  1535 ;                R2-R0   Scratch
                     0618  1536 ;
                     0618  1537 ;  OUTPUTS:      R8,R7   Garbage
                     0618  1538 ;                R6,R5   Preserved
                     0618  1539 ;                R4-R0   Garbage
                     0618  1540 ;
                     0618  1541 ;-
                     0618  1542 ACT$CONFIRM::                                    ; Connect Confirm or Reject
          35   10    0618  1543          BSBB    SETUP_XWB                      ; Do common setup
                     061A  1544          ;
                     061A  1545          ;
                     061A  1546          ;    If the remote end of the Logical-link is on the local node then
                     061A  1547          ;    use the same "path".  This allows loopbacked lines to be used for
                     061A  1548          ;    all logical-link traffic in both directions -- which seems like a
                     061A  1549          ;    sensible thing to do even though this may be a depature from the
                     061A  1550          ;    Network Management architecture.
                     061A  1551          ;
                     061A  1552          ;
             38 A5   B5   061A  1553          TSTW    XWB$W_PATH(R5)             ; Has a path been chosen ?
                21   12   061D  1554          BNEQ    20$                        ; If NEQ then yes
          52    30 A5   D0   061F  1555          MOVL    XWB$L_VCB(R5),R2           ; Get the RCB
    0E A2    3A A5   B1   0623  1556          CMPW    XWB$W_REMNOD(R5),RCB$W_ADDR(R2) ; Is the remote node us?
                16   12   0628  1557          BNEQ    20$                        ; If NEQ no
                     062A  1558          ;
                38   BB   062A  1559          PUSHR   #^M<R3,R4,R5>              ; Save regs
          53    3C A5   3C   062C  1560          MOVZWL  XWB$W_REMLNK(R5),R3        ; Get remote link i.d.
                0676   30   0630  1561          BSBW    NET$XWB_LOCLNK             ; Find corresponding XWB
             52    55   D0   0633  1562          MOVL    R5,R2                      ; Copy XWB address to new reg
                38   BA   0636  1563          POPR    #^M<R3,R4,R5>              ; Restore regs
                     0638  1564          ;
                05 52   E8   0638  1565          BLBS    R2,20$                     ; If LBS then no XWB was found
    38 A5       38 A2   B0   063B  1566          MOVW    XWB$W_PATH(R2),XWB$W_PATH(R5) ; Use partner's path i.d.
    09 20 A3    08   E1   0640  1567 20$:       BBC     #IO$V_ABORT,IRP$W_FUNC(R3),100$ ; If BC then not connect reject
                     0645  1568          ASSUME  NET$C_DR_NORMAL  EQ  0
                46 A5   B4   0645  1569          CLRW    XWB$W_X_REASON(R5)         ; Setup disconnect reason code
             50    01   7D   0648  1570          MOVQ    S^#SS$_NORMAL,R0           ; Setup IOSB value
                007C   30   064B  1571          BSBW    NET$CMPL_ACC               ; Complete the IO$_ACCESS IRP
                     05   064E  1572 100$:      RSB                                ; Done
                     064F  1573
                     064F  1574 ACT$INITIATE::                                   ; Connect Initiate request
```

```
                        064F  1575 SETUP_XWB:                                        ; Setup common fields
                  0C A5  B6  064F  1576          INCW    XWB$W_REFCNT(R5)          ; Account for accessor
          016C C5   34 A5  D0  0652  1577          MOVL    XWB$L_PID(R5),XWB$$+ACB$L_PID(R5) ; Setup Special Kernal AST PID
             0080 C5    53  D0  0658  1578          MOVL    R3,XWB$L_IRP_ACC(R5)      ; Store IRP address
                     58 A6  D4  065D  1579          CLRL    UCB$L_IRP(R6)             ; Clear IRP pointer to prevent
                        0660  1580                                                    ; immediate I/O completion
                10 A5    56  D0  0660  1581          MOVL    R6,XWB$L_ORGUCB(R5)       ; Setup UCB ptr
                54   4C A3  D0  0664  1582          MOVL    IRP$L_DIAGBUF(R3),R4      ; Get ICB ptr
                   4C A3  D4  0668  1583          CLRL    IRP$L_DIAGBUF(R3)         ; Dettach it from IRP
             010C C5    54  D0  066B  1584          MOVL    R4,XWB$L_ICB(R5)          ; Attach it to XWB
          1C A5    0100 8F  A8  0670  1585          BISW    #XWB$M_FLG_SCD,XWB$W_FLG(R5) ; Set send message flag
                        0676  1586 30$:    ;
                        0676  1587         ;
                        0676  1588         ;    Setup pre-allocated byte quota to take upon entering the RUN state
                        0676  1589         ;
                        0676  1590         ;
                        0676  1591 ;&      CLRW    XWB$W_X_QUO(R5)           ; Pre-allocate none for rcv's
                        0676  1592 ;&      CLRW    XWB$W_R_QUO(R5)           ; Pre-allocate none for rcv's
                        0676  1593         ;
                        0676  1594         ;
                        0676  1595         ;    Move remainder of parameters from the ICB
                        0676  1596         ;
                        0676  1597         ;
                     38 BB  0676  1598          PUSHR   #^M<R3,R4,R5>                     ; Save MOVC regs
                        0678  1599                                                    ;
                51   7C A4  9E  0678  1600          MOVAB   ICB$B_DATA(R4),R1         ; Get source pointer
                     06E0  30  067C  1601          BSBW    NET$MOV_TO_XWB            ; Move counted string
                        067F  1602                                                    ; to XWB$B_DATA...
                        067F  1603          ASSUME  ICB$C_RID  LE  XWB$C_RID          ;
             6F A5    0092 C4  90  067F  1604          MOVB    ICB$B_RID(R4),XWB$B_RID(R5) ; Move the count field
                        0685  1605                                                    ;
          20    0093 C4    10  2C  0685  1606          MOVC5   #ICB$C_RID,ICB$T_RID(R4),#^A' ',- ; Move the remote
                70 A5    10      068B  1607                  #XWB$C_RID,XWB$T_RID(R5)  ; i.d. text
                        068E  1608                                                    ;
                     38 BA  068E  1609          POPR    #^M<R3,R4,R5>             ; Restore regs
                50   0C A4  B0  0690  1610          MOVW    ICB$W_RETRAN(R4),R0       ; Get rexmt factor
                     04 15  0694  1611          BLEQ    50$                       ; If LEQ keep default
                54 A5    50  B0  0696  1612          MOVW    R0,                 XWB$W_RETRAN(R5)  ; Update rexmt factor
                   52 A5  B4  069A  1613 50$:    CLRW                        XWB$W_PROGRESS(R5) ; Init progress count
                38 A5    64  B0  069D  1614          MOVW    ICB$W_PATH(R4),     XWB$W_PATH(R5)  ; Circuit to use
             40 A5    12 A4  B0  06A1  1615          MOVW    ICB$W_SEGSIZ(R4),   XWB$W_LOCSIZ(R5) ; Rcv buffer size
             56 A5    0E A4  B0  06A6  1616          MOVW    ICB$W_DLY_FACT(R4), XWB$W_DLY_FACT(R5) ; Delay factor
             58 A5    10 A4  B0  06AB  1617          MOVW    ICB$W_DLY_WGHT(R4), XWB$W_DLY_WGHT(R5) ; Delay weight
             4C A5    06 A4  B0  06B0  1618          MOVW    ICB$W_TIM_INACT(R4),XWB$W_TIM_INACT(R5) ; Inactivity timer
                        06B5  1619         ;
                        06B5  1620         ;
                        06B5  1621         ;    Setup TIMER and DELAY so that the Connect message will be
                        06B5  1622         ;    retransmitted periodically if necessary.  This is done by choosing
                        06B5  1623         ;    a DELAY which will allow RETRAN retransmission before the amount
                        06B5  1624         ;    of time left in TIMER expires.
                        06B5  1625         ;
                        06B5  1626         ;
                54 A5    A7  06B5  1627          DIVW3   XWB$W_RETRAN(R5),-        ; TIMER has number of ticks
             4E A5      50 A5      06B8  1628                  XWB$W_TIMER(R5),XWB$W_DELAY(R5) ; left before timeout
                56 A5    A6  06BC  1629          DIVW    XWB$W_DLY_FACT(R5),-      ; Adjust for the "delay factor"
                   4E A5      06BF  1630                  XWB$W_DELAY(R5)           ;
                     03 12  06C1  1631          BNEQ    70$                       ; If NEQ then use it
```

```
4E A5  B6  06C3  1632
 F937' 30  06C6  1633 70$:    BSBW    NET$RESET_TIMER                 ; Reset XWB$W_TIMER
       05  06C9  1634          RSB                                     ; Done
           06CA  1635
```

```
                        06CA  1637 .SBTTL  NET$CMPL_ACC    - Complete IO$_ACCESS, fill in window
                        06CA  1638 ;++
                        06CA  1639 ;
                        06CA  1640 ;  The access function currently being processed is completed.
                        06CA  1641 ;  If the I/O completion status is not successful then the window of the
                        06CA  1642 ;  channel associated with the IRP is cleared.
                        06CA  1643 ;
                        06CA  1644 ;
                        06CA  1645 ;  INPUTS:  R5  XWB address
                        06CA  1646 ;           R1  Second IOSB longword value
                        06CA  1647 ;           R0  First  IOSB longword value
                        06CA  1648 ;
                        06CA  1649 ;  OUTPUTS: R1  Garbage
                        06CA  1650 ;           R0  SS$_NORMAL
                        06CA  1651 ;
                        06CA  1652 ;           All other registers are preserved.
                        06CA  1653 ;
                        06CA  1654 ;-
                        06CA  1655 NET$CMPL_ACC::                               ; Complete access, fill in window
    01BC 8F     BB      06CA  1656          PUSHR   #^M<R2,R3,R4,R5,R7,R8>      ; Save regs
                        06CE  1657                                              ;
04    4E A5     B1      06CE  1658          CMPW    XWB$W_DELAY(R5),#4          ; Make sure initial "delay" estimate
          04    1E      06D2  1659          BGEQU   30$                         ; is at least 4 seconds
  4E A5   04    B0      06D4  1660          MOVW    #4,XWB$W_DELAY(R5)          ;
53  0080 C5     D0      06D8  1661 30$:     MOVL    XWB$L_IRP_ACC(R5),R3        ; Get IO$_ACCESS IRP
          24    13      06DD  1662          BEQL    200$                        ; If EQL then none
    0080 C5     D4      06DF  1663          CLRL    XWB$L_IRP_ACC(R5)           ; Remove IRP
 38 A3    50    7D      06E3  1664          MOVQ    R0,IRP$L_IOST1(R3)          ; Save I/O status
                        06E7  1665          ;
                        06E7  1666          ;
                        06E7  1667          ;   Setup the CCB$L_WIND value and deallocate the ICB.
                        06E7  1668          ;
                        06E7  1669          ;   If either the access was unsuccessful or the request was for a
                        06E7  1670          ;   Connect Reject, then cleanup from the IO$_ACCESS attempt and
                        06E7  1671          ;   leave the CCB$L_WIND image at zero.
                        06E7  1672          ;
                        06E7  1673          ;
      08    E0      06E7  1674          BBS     #IO$V_ABORT,-               ; If BS, Connect Reject
06 20 A3              06E9  1675                  IRP$W_FUNC(R5),60$          ;
   5B A5    94      06EC  1676          CLRB    XWB$B_DATA(R5)              ; Init optional data cell to prepare
                        06EF  1677                                          ; for eventual disconnect
   05 50    E8      06EF  1678          BLBS    R0,100$                     ; If LBS then successful IO$_ACCESS
    011F    30      06F2  1679 60$:     BSBW    CLEANUP_ACCESS              ; Clean up from access I/O fct
      09    11      06F5  1680          BRB     110$                        ; Complete the I/O
    00F4    30      06F7  1681 100$:    BSBW    GET_WNDSC                   ; Get CCB$L_WIND image descriptor
67    55    D0      06FA  1682          MOVL    R5,(R7)                     ; Setup CCB$L_WIND value
    0161    30      06FD  1683          BSBW    DEAL_ICB                    ; Deallocate the ICB
                        0700  1684 110$:    ;
                        0700  1685          ;
                        0700  1686          ;   Complete the I/O
                        0700  1687          ;
                        0700  1688          ;
    067D    30      0700  1689          BSBW    NET$POST_IO                 ; Post IRP for completion
                        0703  1690                                          ;
01BC 8F     BA      0703  1691 200$:    POPR    #^M<R2,R3,R4,R5,R7,R8>      ; Restore regs
50    01    D0      0707  1692          MOVL    S^#SS$_NORMAL,R0            ; Success
                        070A  1693          RSB
```

```
070B  1694
070B  1695
```

```
                     070B  1697 .SBTTL  ACT$ENT_RUN     - Enter RUN state action routine
                     070B  1698 ;+
                     070B  1699 ;
                     070B  1700 ;  This routine is entered to setup the XWB for entering the RUN state.
                     070B  1701 ;
                     070B  1702 ;  INPUTS:  R7  Event code  - it will be reprocessed via the complex event
                     070B  1703 ;                            mechanism.  Note that the state should have
                     070B  1704 ;                            been updated by then.
                     070B  1705 ;           R5  XWB address
                     070B  1706 ;           R0  Scratch
                     070B  1707 ;
                     070B  1708 ;  OUTPUTS: R0 garbage
                     070B  1709 ;
                     070B  1710 ;           All other registers are preserved.
                     070B  1711 ;
                     070B  1712 ;-
                     070B  1713 ACT$ENT_RUN::                                ; Enter RUN state
F8F2'  30            070B  1714          BSBW    NET$SETUP_RUN              ; Setup XWB for RUN state
FC1F   31            070E  1715          BRW     NET$COMPLEX_EV             ; Change state and resignal the event
                     0711  1716
```

```
                     0711  1718 .SBTTL  NET$FDT_DEACCESS- IO$_DEACCESS FDT processing
                     0711  1719 .SBTTL  NET$DEACCESS    - IO$_DEACCESS "startio" processing
                     0711  1720 ;++
                     0711  1721 ;
                     0711  1722 ;  INPUTS:       AP      Pointer to the QIO P1 parameter
                     0711  1723 ;                R8      Must be saved/restored if return to Exec for next
                     0711  1724 ;                        FDT routine
                     0711  1725 ;                R7      I/O function code without modifiers
                     0711  1726 ;                R6      CCB address
                     0711  1727 ;                R5      UCB address
                     0711  1728 ;                R4      PCB address
                     0711  1729 ;                R3      IRP address
                     0711  1730 ;                R2-R0   Scratch
                     0711  1731 ;
                     0711  1732 ;  OUTPUTS:      R5,R3   Preserved
                     0711  1733 ;
                     0711  1734 ;                All other regs may be clobbered.
                     0711  1735 ;
                     0711  1736 ;--
                     0711  1737 NET$FDT_DEACCESS::                                   ; IO$_DEACCESS FDT routine
        4C A3    D4  0711  1738          CLRL    IRP$L_DIAGBUF(R3)                  ; Zero misc buffer pointer
50   00AC 8F    3C  0714  1739          MOVZWL  #SS$_FILNOTACC,R0                  ; Assume "link not accessed"
  18 A3    01    CA  0719  1740          BICL    #1,IRP$L_WIND(R3)                  ; Clear interlock bit
           52    18  071D  1741          BGEQ    20$                                ; If GEQ, link is not accessed
                     071F  1742                                                     ;
      0F78 8F    BB  071F  1743          PUSHR   #^M<R3,R4,R5,R6,R8,R9,R10,R11>     ; Save regs
                     0723  1744          DSBINT  UCB$B_FIPL(R5)                     ; Synchronize
      5B   01    D0  072A  1745          MOVL    #1,R11                             ; Say "okay to go to IPL 2"
                     072D  1746                                                     ;
   55   18 A3    D0  072D  1747          MOVL    IRP$L_WIND(R3),R5                  ; Switch to XWB context
                     0731  1748          ;
                     0731  1749          ;
                     0731  1750          ;    Setup disconnect reasons codes as appropriate
                     0731  1751          ;
                     0731  1752          ;
44 A5    0064 8F    B1  0731  1753          CMPW    #NET$C_DR_INVALID,XWB$W_R_REASON(R5) ; Rcv'd reason code yet ?
           06    12  0737  1754          BNEQ    10$                                ; If NEQ yes
44 A5    0066 8F    3C  0739  1755          MOVZWL  #NET$C_DR_DEACC,  XWB$W_R_REASON(R5) ; Setup local reason
46 A5    0064 8F    B1  073F  1756 10$:     CMPW    #NET$C_DR_INVALID,XWB$W_X_REASON(R5) ; Xmt reason code setup ?
           0D    1A  0745  1757          BGTRU   20$                                ; If GTRU, yes
   46 A5    00    3C  0747  1758          MOVZWL  #NET$C_DR_NORMAL, XWB$W_X_REASON(R5) ; Assume ordinary disconn.
04 20 A3    08    E1  074B  1759          BBC     #IO$V_ABORT,      IRP$W_FUNC(R3),20$ ; If BS, must abort all I/O
   46 A5    09    3C  0750  1760          MOVZWL  #NET$C_DR_ABORT,  XWB$W_X_REASON(R5) ; Else, set "disc. abort"
                     0754  1761 20$:     ;
                     0754  1762          ;
                     0754  1763          ;    If IO$V_ABORT is set, then both the transmitter and receiver must
                     0754  1764          ;    be run-down.  Otherwise, this is a "synchronous" disconnect and
                     0754  1765          ;    the transmitter must be allowed to send all data before breaking
                     0754  1766          ;    the link.
                     0754  1767          ;
                     0754  1768          ;
0B 0E A5    04    E0  0754  1769          BBS     #XWB$V_STS_CON,   XWB$W_STS(R5),100$ ; If BS, not in RUN format
03 20 A3    08    E1  0759  1770          BBC     #IO$V_ABORT,      IRP$W_FUNC(R3),50$ ; If BS, must abort all I/O
         0161    30  075E  1771          BSBW    DRAIN_XMT                          ; Run-down the transmitter
         01BD    30  0761  1772 50$:     BSBW    DRAIN_RCV                          ; Run-down the receiver
                     0764  1773                                                     ;
                     0764  1774 100$:    ENBINT                                     ; Restore IPL
```

```
                OF78 8F     BA   0767  1775          POPR    #^M<R3,R4,R5,R6,R8,R9,R10,R11>  ; Restore regs
                            0768  1776                                                        ;
00000000'GF     17          076B  1777          JMP     G^ACP$DEACCESS                      ; Goto system FDT routine
00000000'GF     17          0771  1778 200$:     JMP     G^EXE$ABORTIO                       ; Abort the I/O
                            0777  1779
                            0777  1780
                            0777  1781 NET$DEACCESS::                                       ; User QIO to break link
                  0074      30    0777  1782          BSBW    GET_WNDSC                     ; Get CCB$L_WIND image desc
                    67      D4    077A  1783          CLRL    (R7)                          ; Clear CCB$L_WIND image
                38 A3       01    3C    077C  1784          MOVZWL  #SS$_NORMAL,IRP$L_IOST1(R3)   ; Setup success status
                2A A3       02    A8    0780  1785          BISW    #IRP$M_FUNC,IRP$W_STS(R3)     ; Mark for write back
                32 A3       01    D0    0784  1786          MOVL    #1,IRP$L_BCNT(R3)             ; Write back 1 (the window) ABD
                    57      12    D0    0788  1787          MOVL    #NETEVT$_DEA,R7               ; Setup event code in case R5
                            05    078B  1788          RSB                                   ; is a valid XWB pointer
                                  078C  1789
                                  078C  1790
                                  078C  1791 ACT$DEACCESS::                                 ; User QIO to break link
                  0085      30    078C  1792          BSBW    CLEANUP_ACCESS                ; Clean up from access I/O fct
                  0065      30    078F  1793          BSBW    GET_P2DSC                     ; Get optional data descriptor
                    58      D7    0792  1794          DECL    R8                            ; Reduce length by count field
                    1C      19    0794  1795          BLSS    ACT$RES_DISC                  ; If LSS, then no data
                51  67      9A    0796  1796          MOVZBL  (R7),R1                       ; Get count value from string
                51  58      D1    0799  1797          CMPL    R8,R1                         ; Take minimum of size from
                    03      1F    079C  1798          BLSSU   20$                           ;  descriptor and size from
                58  51      D0    079E  1799          MOVL    R1,R8                         ;  string
                10  58      D1    07A1  1800 20$:     CMPL    R8,#16                        ; Take minimum of what's there
                    03      1F    07A4  1801          BLSSU   30$                           ;  and max allowed by NSP
                58  10      D0    07A6  1802          MOVL    #16,R8                        ;
                67  58      90    07A9  1803 30$:     MOVB    R8,(R7)                       ; Setup count field in string
                51  57      D0    07AC  1804          MOVL    R7,R1                         ; Setup source ptr
                  05AD      30    07AF  1805          BSBW    NET$MOV_TO_XWB                ; Move counted string to
                            07B2  1806                                                      ; XWB$B_DATA
                            07B2  1807 ACT$RES_DISC::                                       ; Resume Disconnect processing
                            07B2  1808          ;
                            07B2  1809          ;
                            07B2  1810          ;   If the XWB is idle, continue processing this event.  Else, dismiss
                            07B2  1811          ;   this event for now and resume it when the XWB goes idle.  This is
                            07B2  1812          ;   the only way to do a "synchronous disconnect" with NSP pipelining
                            07B2  1813          ;   causing user Transmit IRP's to be completed before the CXB's are
                            07B2  1814          ;   actually transmitted.
                            07B2  1815          ;
                            07B2  1816          ;
                      0E    10    07B2  1817          BSBB    NET$CHK_X_IDLE                ; Is the transmitter idle ?
                04  50      E9    07B4  1818          BLBC    R0,100$                       ; If LBS then no
                  00F5      30    07B7  1819          BSBW    NET$PURG_RUN                  ; Cleanup if necessary
                            05    07BA  1820          RSB                                   ; Return to change state
                            07BB  1821                                                      ;
             0E A5      08    A8    07BB  1822 100$:    BISW    #XWB$M_STS_DIS,XWB$W_STS(R5)  ; Mark disconnect pending
                  FB68      31    07BF  1823          BRW     NET$END_EVENT                 ; Dismiss this event for now
                            07C2  1824
```

```
                        07C2  1826
                        07C2  1827                      .ENABL  LSB
                        07C2  1828
                        07C2  1829 NET$CHK_X_IDLE::                                     ; See if transmitter is idle
20 0E A5    04    E0    07C2  1830                BBS     #XWB$V_STS_CON,XWB$W_STS(R5),10$; If BS, not in RUN format
 50   00BC C5    D0    07C7  1831                MOVL     XWB$T_DT+LSB$L_X_CXB(R5),R0    ; Get next DATA CXB
 50   00EC C5    C8    07CC  1832                BISL     XWB$T_LI+LSB$L_X_CXB(R5),R0    ;  OR in next Interrupt CXB
                        07D1  1833 CHK_X_IRP:                                           ; Check for Xmt IRP's
 50   00B4 C5    C8    07D1  1834                BISL     XWB$T_DT+LSB$L_X_PND(R5),R0    ;  OR in pending DATA Xmt IRP's
 50   00B8 C5    C8    07D6  1835                BISL     XWB$T_DT+LSB$L_X_IRP(R5),R0    ;  OR in spent DATA Xmt IPR's
 50   00E4 C5    C8    07DB  1836                BISL     XWB$T_LI+LSB$L_X_PND(R5),R0    ;  OR in pending Int. Xmt IRP's
 50   00B8 C5    C8    07E0  1837                BISL     XWB$T_DT+LSB$L_X_IRP(R5),R0    ;  OR in spent Int. Xmt IPR's
            04    12    07E5  1838                BNEQ     20$                            ; If NEQ then not idle
      50    01    D0    07E7  1839 10$:           MOVL    #1,R0                           ; Say "idle"
            05    07EA  1840                      RSB                                     ; Done
                  07EB  1841                                                              ;
            50    D4    07EB  1842 20$:           CLRL     R0                             ; Say "non-idle"
            05    07ED  1843                      RSB                                     ; Done
                  07EE  1844
                  07EE  1845                      .DSABL  LSB
                  07EE  1846
                  07EE  1847
                  07EE  1848                      .ENABL  LSB
                  07EE  1849 GET_WNDSC:                                   ; Get window descriptor
            58    D4    07EE  1850                CLRL    R8                ; Get descriptor offset
            12    11    07F0  1851                BRB     10$               ; Continue
                  07F2  1852 GET_P1DSC:                                   ; Get P1 descriptor
      58    08    D0    07F2  1853                MOVL    #8,R8             ; Get descriptor offset
            0D    11    07F5  1854                BRB     10$               ; Continue
                  07F7  1855 GET_P2DSC:                                   ; Get P2 descriptor
      58    10    D0    07F7  1856                MOVL    #8*2,R8           ; Get descriptor offset
            08    11    07FA  1857                BRB     10$               ; Continue in common
                  07FC  1858 GET_P3DSC:                                   ; Get P3 descriptor
      58    18    D0    07FC  1859                MOVL    #8*3,R8           ; Get descriptor offset
            03    11    07FF  1860                BRB     10$               ; Continue in common
                  0801  1861 GET_P4DSC:                                   ; Get P4 descriptor
      58    20    D0    0801  1862                MOVL    #8*4,R8           ; Get desciptor offset
   58     2C B3    C0    0804  1863 10$:           ADDL    @IRP$L_SVAPTE(R3),R8      ; Get descriptor address
      57    88    3C    0808  1864                MOVZWL  (R8)+,R7                  ; Get offset to data
57    FF A847    9E    080B  1865                MOVAB   -1(R8)[R7],R7             ; Get ptr to data after skipping
                  0810  1866                                                        ; over access mode byte
      58    68    3C    0810  1867                MOVZWL  (R8),R8                   ; Get length of data
            05    0813  1868                      RSB
                  0814  1869                      .DSABL  LSB
                  0814  1870
```

```
                                0814  1872 .SBTTL  CLEANUP_ACCESS  - Cleanup XWB for terminated IO$_ACCESS
                                0814  1873 ;+
                                0814  1874 ;
                                0814  1875 ;  INPUTS:      R5 - XWB address, low bit set if none
                                0814  1876 ;               R3 - IRP address
                                0814  1877 ;
                                0814  1878 ;  OUTPUTS:     All registers are preserved.
                                0814  1879 ;
                                0814  1880 ;
                                0814  1881 ;-
                                0814  1882 CLEANUP_ACCESS:
                          07    BB    0814  1883          PUSHR   #^M<R0,R1,R2>                       ; Save regs
                                0816  1884                                                          ;
                          52    D4    0816  1885          CLRL    R2                                  ; Assume no byte quota to return
                       19 55    E8    0818  1886          BLBS    R5,20$                              ; If LBS then no XWB
                          44    10    081B  1887          BSBB    DEAL_ICB                            ; Deallocate the ICB
                        04F6    30    081D  1888          BSBW    NET$DRAIN_FREE_CXB                  ; Drain CXB free queue
                      0C A5     B7    0820  1889          DECW    XWB$W_REFCNT(R5)                    ; Account for loss of accessor
                          38    12    0823  1890          BNEQ    200$                                ; Br if last accessor
                      10 A5     D4    0825  1891 10$:     CLRL    XWB$L_ORGUCB(R5)                    ; XWB is unowned
                      34 A5     D4    0828  1892          CLRL    XWB$L_PID(R5)                       ; XWB is unowned
                04 0E A5        0A    E0    082B  1893          BBS     #XWB$V_STS_ASTPND,XWB$W_STS(R5),20$ ; If BS, AST block in use
                    016C C5     D4    0830  1894          CLRL    XWB$$_ACB$L_PID(R5)                 ; Prevent false AST delivery
                                0834  1895 20$:     ;
                                0834  1896          ;
                                0834  1897          ;    Return BYTCNT and FILCNT quota
                                0834  1898          ;
                                0834  1899          ;
                      50    0C A3   3C    0834  1900          MOVZWL  IRP$L_PID(R3),R0                    ; Get PID index
                51    00000000'GF   D0    0838  1901          MOVL    G^SCH$GL_PCBVEC,R1                  ; Get PCB vector address
                        51    6140  D0    083F  1902          MOVL    (R1)[R0],R1                         ; Get PCB address
                   0C A3      60 A1  D1    0843  1903          CMPL    PCB$L_PID(R1),IRP$L_PID(R3)         ; Still there ?
                              10    12    0848  1904          BNEQ    30$                                 ; If not branch
                     50   0080 C1    D0    084A  1905          MOVL    PCB$L_JIB(R1),R0                    ; Get JIB
                             30 A0   B6    084F  1906          INCW    JIB$W_FILCNT(R0)                    ; Return quota for IO$_ACCESS
                        20 A0   52    C0    0852  1907          ADDL    R2,JIB$L_BYTCNT(R0)                 ; Return byte quota
                        24 A0   52    C0    0856  1908          ADDL    R2,JIB$L_BYTLM(R0)                  ; Here too
                                085A  1909                                                          ;
                          07    BA    085A  1910 30$:     POPR    #^M<R0,R1,R2>                       ; Restore regs
                                05    085C  1911          RSB                                         ; Done
                                085D  1912
                                085D  1913 200$:    BUG_CHECK NETNOSTATE,FATAL                      ; Invalid reference count
                                0861  1914
                                0861  1915 DEAL_ICB:                                               ; Deallocate the ICB
                          7E    50    7D    0861  1916          MOVQ    R0,-(SP)                    ; Save regs
                                0864  1917                                                          ;
                0E 0E A5        04    E1    0864  1918          BBC     #XWB$V_STS_CON,XWB$W_STS(R5),40$ ; If BC, XWB$L_ICB is invalid
                     50   010C C5    D0    0869  1919          MOVL    XWB$L_ICB(R5),R0                ; Get buffer for deallocation
                              03    18    086E  1920          BGEQ    30$                             ; If GEQ then none
                          04DB      30    0870  1921          BSBW    NET$DEALLOCATE                  ; Deallocate block in R0
                          010C C5   D4    0873  1922 30$:     CLRL    XWB$L_ICB(R5)                   ; Say "no ICB"
                                0877  1923                                                          ;
                          50    8E    7D    0877  1924 40$:     MOVQ    (SP)+,R0                    ; Restore regs
                                05    087A  1925          RSB                                         ; Done
                                087B  1926
```

```
                                                H 15
                     087B  1928 .SBTTL  NET$CANCEL      - Cancel I/O routine
                     087B  1929 ;+
                     087B  1930 ;
                     087B  1931 ;  Most of the work for the Cancel-I/O sequence will occur when the special
                     087B  1932 ;  IO$_ACPCONTROL QIO is issued by the $CANCEL system service.
                     087B  1933 ;
                     087B  1934 ;  In all cases, the ACP is informed via a mailbox message since special
                     087B  1935 ;  cleanup may be needed in the ACP (e.g. declared name cleanup).  Note that
                     087B  1936 ;  the special Cancel IRP is only sent to the ACP if there is a logical-link
                     087B  1937 ;  active.
                     087B  1938 ;
                     087B  1939 ;
                     087B  1940 ;  INPUTS:       R5      UCB address
                     087B  1941 ;                R4      PCB address
                     087B  1942 ;                R3      IRP address if UCB is busy
                     087B  1943 ;                R2      Channel number
                     087B  1944 ;                R1,R0   Scratch
                     087B  1945 ;
                     087B  1946 ;                NET$C_IPL
                     087B  1947 ;
                     087B  1948 ;  OUTPUTS:      R3-R0   Garbage
                     087B  1949 ;
                     087B  1950 ;                All other registers are preserved
                     087B  1951 ;
                     087B  1952 ;-
                     087B  1953 NET$CANCEL:                                          ; Cancel I/O entry point
 50   34 A5   D0     087B  1954          MOVL    UCB$L_VCB(R5),R0                   ; Get VCB address
        24   13      087F  1955          BEQL    50$                                ; If EQL then none
                     0881  1956          ;
                     0881  1957          ;
                     0881  1958          ;    Tell the ACP
                     0881  1959          ;
                     0881  1960          ;
    01BC 8F    BB    0881  1961          PUSHR   #^M<R2,R3,R4,R5,R7,R8>             ; Save regs
                     0885  1962          ;
    57   52    D0    0885  1963          MOVL    R2,R7                              ; Save channel number
55    14 A0    D0    0888  1964          MOVL    RCB$L_ACP_UCB(R0),R5               ; Get the ACP's UCB
    58   36    3C    088C  1965          MOVZWL  #MSG$_PATHLOST,R8                  ; Setup mailbox message code
    52   06    D0    088F  1966          MOVL    #6,R2                              ; No. of bytes to be entered
      02DC     30    0892  1967          BSBW    NET$SEND_MBX                       ; Setup the message
      09 50    E9    0895  1968          BLBC    R0,30$                             ; Br on error -- ignore it
83    60 A4    D0    0898  1969          MOVL    PCB$L_PID(R4),(R3)+                ; Enter the PID
    83   57    B0    089C  1970          MOVW    R7,(R3)+                           ; Enter channel
         9E    16    089F  1971          JSB     @(SP)+                             ; Send the message
                     08A1  1972          ;
    01BC 8F    BA    08A1  1973 30$:     POPR    #^M<R2,R3,R4,R5,R7,R8>             ; Restore regs
                     08A5  1974 50$:     ;
                     08A5  1975          ;
                     08A5  1976          ;    If the unit is busy then it must be a bug sinc NET$STARTIO never
                     08A5  1977          ;    allows an I/O queue to build on the UCB
                     08A5  1978          ;
                     08A5  1979          ;
01 64 A5    08   E0  08A5  1980          BBS     #UCB$V_BSY,UCB$W_STS(R5),100$      ; Done if UCB is not busy
               05    08AA  1981          RSB                                        ; Done
                     08AB  1982          ;
                     08AB  1983 100$:    BUG_CHECK NETNOSTATE,FATAL                 ; Our UCB assumptions are wrong
                     08AF  1984
```

```
                                08AF  1986 .SBTTL  NET$PURG_RUN    - Cleanup XWB to exit RUN state
                                08AF  1987 ;+
                                08AF  1988 ;
                                08AF  1989 ;  The receiver and transmitter are run-down on both the DATA and INT/LS
                                08AF  1990 ;  LSB's.
                                08AF  1991 ;
                                08AF  1992 ;  It is assumed that this routine is called as a result of a call from one
                                08AF  1993 ;  of the state transition action routines and that there will be a state
                                08AF  1994 ;  transition out of the RUN state as the event processing is completed.  This
                                08AF  1995 ;  is because certain processing -- such as the setting and clearing of XWB
                                08AF  1996 ;  flags -- is assumed to be done as part of the state transition processing
                                08AF  1997 ;  and is therefore done by this routine.
                                08AF  1998 ;
                                08AF  1999 ;
                                08AF  2000 ;  INPUTS:       R5      XWB address; low bit set if no XWB
                                08AF  2001 ;                R0      Scratch
                                08AF  2002 ;
                                08AF  2003 ;  OUTPUTS:      R0      Garbage
                                08AF  2004 ;
                                08AF  2005 ;                All other registers are preserved
                                08AF  2006 ;
                                08AF  2007 ;-
                                08AF  2008 NET$PURG_RUN::                                   ; Leave the RUN state
                      01DE 8F   BB  08AF  2009          PUSHR  #^M<R1,R2,R3,R4,R6,R7,R8>    ; Save regs
                                08B3  2010                                                  ;
          05 0E A5    04        E0  08B3  2011          BBS    #XWB$V_STS_CON,XWB$W_STS(R5),20$ ; If BS, not in RUN format
                        08        10  08B8  2012          BSBB    DRAIN_XMT                   ; Drain the transmitter
                      0064        30  08BA  2013          BSBW    DRAIN_RCV                   ; Drain the receiver
                                08BD  2014                                                  ;
                      01DE 8F   BA  08BD  2015 20$:       POPR   #^M<R1,R2,R3,R4,R6,R7,R8>    ; Restore regs
                                05  08C1  2016          RSB
                                08C2  2017
                                08C2  2018
                                08C2  2019 DRAIN_XMT:                                       ; Drain the xmitter
                                08C2  2020          ;
                                08C2  2021          ;    All transmit CXB's are detached and eventually deallocated.
                                08C2  2022          ;    All transmit IRP's are sent to I/O Post with disconenct status.
                                08C2  2023          ;    The LSB transmitter state variables are updated to reflect an
                                08C2  2024          ;    idle transmitter.
                                08C2  2025          ;
                                08C2  2026          ;
                                08C2  2027          ;    Inputs:    R8,R7   Scratch
                                08C2  2028          ;               R5      XWB address
                                08C2  2029          ;               R4-R0   Scratch
                                08C2  2030          ;
                                08C2  2031          ;    Outputs:   R8,R4-R0  garbage.
                                08C2  2032          ;
                                08C2  2033          ;               All other registers are preserved.
                                08C2  2034          ;
                                08C2  2035          ;
                          F9FB  30  08C2  2036          BSBW    NET$MAP_R_REASON            ; Map disconnect reason to status
                50        02 A0   3C  08C5  2037          MOVZWL  REASON_@_SS(R0),R0          ; Get proper I/O status code
                             51   D4  08C9  2038          CLRL    R1                          ; IOSB second longword
          58    00D4 C5    9E  08CB  2039          MOVAB   XWB$T_LI(R5),R8             ; Get the LS/INT LSB
                             11  10  08D0  2040          BSBB    10$                         ; Do it
          58    00A4 C5    9E  08D2  2041          MOVAB   XWB$T_DT(R5),R8             ; Get the DATA LSB
                             0A  10  08D7  2042          BSBB    10$                         ; Do it
```

```
                        08D9  2043          ;
                        08D9  2044          ;
                        08D9  2045          ;   Simulate an ACK on each active CXB thus causing them to be
                        08D9  2046          ;   deallocated.  This will lead to a false value for LSB$B_X_REQ,
                        08D9  2047          ;   but this is tolerable since we're about to exit the RUN state.
                        08D9  2048          ;
                        08D9  2049          ;      (only the DATA subchannel has CXB's attached to the LSB).
                        08D9  2050          ;
                        08D9  2051          ;
54   0D A8   9A         08D9  2052          MOVZBL  LSB$B_X_CXBACT(R8),R4       ; Number of active Xmt CXB's
       03    13         08DD  2053          BEQL    5$                          ; If EQL then none
     F71E'   30         08DF  2054          BSBW    NET$ACK_XMT_SEGS            ; "ACK" each segment release CXB's
             05         08E2  2055 5$:      RSB                                 ; Done
                        08E3  2056          ;
                        08E3  2057          ;
   7E   50   7D         08E3  2058 10$:     MOVQ    R0,-(SP)                    ; Save IOSB image
                        08E6  2059          ;
                        08E6  2060          ;
                        08E6  2061          ;   Update Xmitter state variables
                        08E6  2062          ;
                        08E6  2063          ;
02 A8   68   B0         08E6  2064          MOVW    LSB$W_LUX(R8),LSB$W_LNX(R8) ; Pretend we've sent all packets
06 A8   68   B0         08EA  2065          MOVW    LSB$W_LUX(R8),LSB$W_HAR(R8) ; Pretend all packets were ACK'd
08 A8   68   B0         08EE  2066          MOVW    LSB$W_LUX(R8),LSB$W_HAA(R8) ; No further ACK's expected
04 A8   68   B0         08F2  2067          MOVW    LSB$W_LUX(R8),LSB$W_HXS(R8) ; No further packets to send
                        08F6  2068          ;
                        08F6  2069          ;
                        08F6  2070          ;   Join the Xmitter's IRP lists, setup each IRP with new I/O status
                        08F6  2071          ;
                        08F6  2072          ;
                        08F6  2073          ASSUME  IRP$L_IOQFL  EQ  0
                        08F6  2074          ;
51   14 A8   9E         08F6  2075          MOVAB   LSB$L_X_IRP(R8),R1          ; Get spent IRP listhead
       0D    11         08FA  2076          BRB     40$
     10 A8   D4         08FC  2077 20$:     CLRL    LSB$L_X_PND(R8)             ; Detach pending IRP list
   61   50   D0         08FF  2078          MOVL    R0,(R1)                     ; Attach it to end of spent IRP list
   51   50   D0         0902  2079 30$:     MOVL    R0,R1                       ; Update last IRP pointer
38 A0   6E   7D         0905  2080          MOVQ    (SP),IRP$L_IOST1(R0)        ; Overwrite status
   50   61   D0         0909  2081 40$:     MOVL    (R1),R0                     ; Get next IRP
        F4   12         090C  2082          BNEQ    30$                         ; If NEQ, IRP was found
50   10 A8   D0         090E  2083          MOVL    LSB$L_X_PND(R8),R0          ; Get pending IRP list
        E8   12         0912  2084          BNEQ    20$                         ; If NEQ, not empty
53   14 A8   D0         0914  2085          MOVL    LSB$L_X_IRP(R8),R3          ; Get first IRP
       03    13         0918  2086          BEQL    100$                        ; If EQL, none
     F6E3'   30         091A  2087          BSBW    NET$XMT_DONE                ; Complete all Xmt IRPs
                        091D  2088          ;
   50   8E   7D         091D  2089 100$:    MOVQ    (SP)+,R0                    ; Restore stack and R0
             05         0920  2090          RSB                                 ; Done
                        0921  2091
                        0921  2092
                        0921  2093 DRAIN_RCV:                                   ; Drain the receiver
                        0921  2094          ;
                        0921  2095          ;   All receive CXB's are detached and deallocated.
                        0921  2096          ;
                        0921  2097          ;   All receive IRP's are sent to I/O Post with disconnect status.
                        0921  2098          ;   For each LSB, LSB$B_R_CXBQUO is zeroed to prevent further CXB's
                        0921  2099          ;   from being received.
```

```
                        0921  2100          ;
                        0921  2101          ;
                        0921  2102          ;   Inputs:     R8      Scratch
                        0921  2103          ;               R5      XWB address
                        0921  2104          ;               R3      Scratch
                        0921  2105          ;               R1-R0   Scratch
                        0921  2106          ;
                        0921  2107          ;   Outputs:    R8,R3,R1,R0  garbage.
                        0921  2108          ;
                        0921  2109          ;               All other registers are preserved.
                        0921  2110          ;
                        0921  2111          ;
        F99C    30      0921  2112          BSBW    NET$MAP_R_REASON            ; Map disconnect reason to status
  50    02 A0   3C      0924  2113          MOVZWL  REASON_W_SS(R0),R0          ; Get proper I/O status code
          51    D4      0928  2114          CLRL    R1                          ; IOSB second longword
  58    00D4 C5 9E      092A  2115          MOVAB   XWB$T_LI(R5),R8             ; Get the LS/INT LSB
          05    10      092F  2116          BSBB    10$                         ; Do it
  58    00A4 C5 9E      0931  2117          MOVAB   XWB$T_DT(R5),R8             ; Get the DATA LSB
     7E   50    7D      0936  2118 10$:     MOVQ    R0,-(SP)                    ; Save IOSB image
                        0939  2119          ;
                        0939  2120          ;
                        0939  2121          ;   Drain Receive CXB list
                        0939  2122          ;
                        0939  2123          ;
        29 A8   94      0939  2124          CLRB    LSB$B_R_CXBQUO(R8)          ; Prevent further receives
        28 A8   94      093C  2125          CLRB    LSB$B_R_CXBCNT(R8)          ; Zero the CXB  in use count
  50    20 A8   D0      093F  2126          MOVL    LSB$L_R_CXB(R8),R0          ; Get first CXB in list
          0E    13      0943  2127          BEQL    40$                         ; If EQL then none
        20 A8   D4      0945  2128          CLRL    LSB$L_R_CXB(R8)             ; Detach entire CXB chain from LSB
                        0948  2129          ;
        10 A0   DD      0948  2130 30$:     PUSHL   CXB$L_LINK(R0)              ; Save ptr to next CXB
         0400   30      094B  2131          BSBW    NET$DEALLOCATE              ; Deallocate block in R0
            50  8ED0    094E  2132          POPL    R0                          ; Get the next CXB
                        0951  2133          ;
           F5   12      0951  2134          BNEQ    30$                         ; If NEQ then loop, else no CXB
                        0953  2135 40$:     ;
                        0953  2136          ;
                        0953  2137          ;   Complete all Rcv IRP's with mapped disconnect status code
                        0953  2138          ;
                        0953  2139          ;
  53    1C A8   D0      0953  2140          MOVL    LSB$L_R_IRP(R8),R3          ; Get next Rcv IRP
          0C    13      0957  2141          BEQL    50$                         ; If EQL then none
        32 A3   B4      0959  2142          CLRW    IRP$W_BCNT(R3)              ; No bytes xferred
  38 A3   6E    7D      095C  2143          MOVQ    (SP),IRP$L_IOST1(R3)        ; Setup I/O status
        F69D'   30      0960  2144          BSBW    NET$RCV_DONE                ; Complete the receive
           EE   11      0963  2145          BRB     40$                         ; Loop
                        0965  2146          ;
  50    8E    7D      0965  2147 50$:     MOVQ    (SP)+,R0                    ; Restore regs
              05      0968  2148          RSB                                 ; Done
                        0969  2149
```

```
                        0969  2151 .SBTTL  NET$ACP_COMM    - Entry for ACP communication
                        0969  2152 ;++
                        0969  2153 ;
                        0969  2154 ;  This routine is called by the ACP for change of status notification
                        0969  2155 ;  including process exit, logical-link "ownership" changes, and datalink
                        0969  2156 ;  transitions.
                        0969  2157 ;
                        0969  2158 ;
                        0969  2159 ;  CALLING SEQUENCE:
                        0969  2160 ;
                        0969  2161 ;         JSB   @CRB$L_INTD+VEC$L_START  at IPL 0
                        0969  2162 ;
                        0969  2163 ;  INPUTS: R5     NET UCB address.
                        0969  2164 ;          R4-R1 Function specific -- see individual action routine preambles
                        0969  2165 ;          R0    Function code as follows:
                        0969  2166 ;
                        0969  2167 ;                 NETUPD$_CONNECT   - Pass NCB to Declared Name mailbox
                        0969  2168 ;                 NETUPD$_PROCRE    - Process created to received connect
                        0969  2169 ;                 NETUPD$_ABORT     - Process couldn't start
                        0969  2170 ;                 NETUPD$_EXIT      - Started process is exiting
                        0969  2171 ;
                        0969  2172 ;                 NETUPD$_DLL_ON    - Datalink has come online - post a receive
                        0969  2173 ;                 NETUPD$_DLL_DLE   - Datalink online for service fcts
                        0969  2174 ;                 NETUPD$_REACT_RCV - Reactivate Datalink receiver
                        0969  2175 ;                 NETUPD$_SEND_HELLO - Force datalink to send a hello message
                        0969  2176 ;
                        0969  2177 ;                 NETUPD$_CRELNK    - Create a logical-link control structure
                        0969  2178 ;                 NETUPD$_DSCLNK    - Graceful disconnect of single link
                        0969  2179 ;                 NETUPD$_ABOLNK    - Force immediate disconnect of all links
                        0969  2180 ;
                        0969  2181 ;                 NETUPD$_BRDCST    - Broadcast mailbox message
                        0969  2182 ;                 NETUPD$_REPLY     - Reply to associated mailbox
                        0969  2183 ;
                        0969  2184 ;  OUTPUTS:  R0 Status
                        0969  2185 ;
                        0969  2186 ;           All other registers are preserved.
                        0969  2187 ;
                        0969  2188 ;--
                0000000C 0969  2189         R3_OFF = 4*3
                00000010 0969  2190         R4_OFF = 4*4
                00000014 0969  2191         R5_OFF = 4*5
                        0969  2192
                        0969  2193                 .ENABL  LSB
                        0969  2194
                        0969  2195 NET$ACP_COMM::                           ; ACP entry point
                        0969  2196         SETIPL  UCB$B_FIPL(R5)           ; Raise IPL to synch access to structures
                        096D  2197
07FF 8F     BB          096D  2198         PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10>  ; Save regs
                        0971  2199
5A   5E     D0          0971  2200         MOVL    SP,R10                   ; Save ptr to saved R0
     0B     10          0974  2201         BSBB    20$                      ; Dispatch on fct code
6E   50     D0          0976  2202         MOVL    R0,(SP)                  ; Overlay return code
                        0979  2203
07FF 8F     BA          0979  2204         POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10>  ; Restore regs
                        097D  2205
                        097D  2206         SETIPL  #0                       ; Restore IPL
                    05  0980  2207         RSB
```

```
                     0981  2208
                     0981  2209 20$:    $DISPATCH  R0,TYPE=B,-          ; Case on function code
                     0981  2210          <-
                     0981  2211              <NETUPD$_CONNECT,   DECLARE>,-  ; Pass NCB to Declared Name mailbox
                     0981  2212              <NETUPD$_PROCRE,    PROCRE>,-  ; Process created to rcv connect
                     0981  2213              <NETUPD$_ABORT,     ABORT>,-   ; Abort single link for given process
                     0981  2214              <NETUPD$_EXIT,      EXIT>,-    ; Started process is exiting
                     0981  2215              -                              ;
                     0981  2216              <NETUPD$_CRELNK,    CRE_LNK>,- ; Create a logical-link
                     0981  2217              <NETUPD$_DSCLNK,    DISC_ONE>,-; Disconnect single logical-link
                     0981  2218              <NETUPD$_ABOLNK,    ABORT_ALL>,-; Abort all logical-links
                     0981  2219              -                              ;
                     0981  2220              <NETUPD$_BRDCST,    BRDCST>,-  ; Broadcast mailbox message
                     0981  2221              <NETUPD$_REPLY,     REPLY>,-   ; Send general mailbox message
                     0981  2222              <NETUPD$_DLL_ON,    DLLTRN>,-  ; Datalink made into "on" state
                     0981  2223              -                              ;
                     0981  2224          >
           018F   31 099B  2225          BRW     UNKNOWN                    ; Let lowest level handle this
                     099E  2226
                     099E  2227 ;+
                     099E  2228 ;  PROCRE   - Process started due to CI received
                     099E  2229 ;
                     099E  2230 ;  INPUTS:  R5  NET UCB address.
                     099E  2231 ;           R4  Scratch
                     099E  2232 ;           R3  Local link number.
                     099E  2233 ;           R2  Scatch
                     099E  2234 ;           R1  PID of process
                     099E  2235 ;
                     099E  2236 ;-
           50   D4   099E  2237 PROCRE: CLRL    R0                         ; Setup for "no PID" match
         009D   30   09A0  2238          BSBW    200$                       ; Get XWB
           11   12   09A3  2239          BNEQ    40$                        ; Done if NEQ
     34 A5   51   D0   09A5  2240          MOVL    R1,XWB$L_PID(R5)           ; Set PID of process allowed
                     09A9  2241                                             ; to complete the connect
           0B   11   09A9  2242          BRB     40$                        ; Done
                     09AB  2243
                     09AB  2244 ;+
                     09AB  2245 ;  ABORT    - Abort single logical-link for a given process
                     09AB  2246 ;
                     09AB  2247 ;  INPUTS:  R5  NET UCB address.
                     09AB  2248 ;           R4  Scratch
                     09AB  2249 ;           R3  Local link number.
                     09AB  2250 ;           R2  Disconnect reason code
                     09AB  2251 ;           R1  PID of process (zero if process not started)
                     09AB  2252 ;
                     09AB  2253 ;-
        50   51   D0   09AB  2254 ABORT:  MOVL    R1,R0                      ; Setup PID (could be zero)
         008F   30   09AE  2255          BSBW    200$                       ; Get XWB
           03   12   09B1  2256          BNEQ    40$                        ; Done if NEQ
         0067   30   09B3  2257          BSBW    180$                       ; Enter DIS state
        50   01   D0   09B6  2258 40$:    MOVL    S^#SS$_NORMAL,R0           ; Report success
                     05   09B9  2259          RSB                        ; Done
                     09BA  2260
                     09BA  2261 ;+
                     09BA  2262 ;  EXIT     - A formerly started process has exited
                     09BA  2263 ;
                     09BA  2264 ;  INPUTS:  R5  NET UCB address
```

```
                     09BA  2265 ;           R4  Scratch
                     09BA  2266 ;           R3  Scratch
                     09BA  2267 ;           R2  Disconnect reason code
                     09BA  2268 ;           R1  PID of process
                     09BA  2269 ;
                     09BA  2270 ;-
55   34 A5   D0      09BA  2271 EXIT:   MOVL    UCB$L_VCB(R5),R5          ; Get RCB
          1B 13      09BE  2272         BEQL    80$                       ; Br if not mounted
55   24 A5   D0      09C0  2273         MOVL    RCB$L_PTR_LTB(R5),R5      ; Get LTB
          15 13      09C4  2274         BEQL    80$                       ; Br if its not there
55   E0 A5   DE      09C6  2275         MOVAL   -XWB$L_LINK -             ;
                     09CA  2276                 +LTB$L_XWB(R5),R5         ; Setup for scan
55   2C A5   D0      09CA  2277 60$:    MOVL    XWB$L_LINK(R5),R5         ; Get next XWB
          0B 13      09CE  2278         BEQL    80$                       ; If EQL then end of list
     50   51   D0      09D0  2279         MOVL    R1,R0                     ; Copy process PID
               73 10      09D3  2280         BSBB    210$                      ; Check process access to XWB via PID
               F3 12      09D5  2281         BNEQ    60$                       ; If NEQ then something wrong
               44 10      09D7  2282         BSBB    180$                      ; Disconnect the link
               EF 11      09D9  2283         BRB     60$                       ; Continue
     50   01   D0      09DB  2284 80$:    MOVL    S^#SS$_NORMAL,R0          ; Success
               05      09DE  2285         RSB                               ; Done
                     09DF  2286
                     09DF  2287
                     09DF  2288 ;+
                     09DF  2289 ;  CRE_LNK  -   Create a single logical-link
                     09DF  2290 ;
                     09DF  2291 ;  INPUTS:  R5  NET UCB address.
                     09DF  2292 ;           R4  Scratch
                     09DF  2293 ;           R3  Logical-link's remote node address
                     09DF  2294 ;           R2  Scratch
                     09DF  2295 ;           R1  PID of process allowed to access link
                     09DF  2296 ;
                     09DF  2297 ;  OUTPUTS:  R0 XWB address, high bit clear => failure code
                     09DF  2298 ;-
                     09DF  2299 CRE_LNK:                                  ; Create single logical-link
          01E4 30    09DF  2300         BSBW    NET$CREATE_XWB            ; Create the structure
        07 50  E9    09E2  2301         BLBC    R0,10$                    ; If LBC, failed
34 A5     51  D0      09E5  2302         MOVL    R1,XWB$L_PID(R5)          ; Setup PID
     50   55   D0      09E9  2303         MOVL    R5,R0                     ; Setup XWB address
               05      09EC  2304 10$:    RSB                               ; Done
                     09ED  2305
                     09ED  2306 ;+
                     09ED  2307 ;  DISC_ONE -   Disconnect a single logical-link
                     09ED  2308 ;
                     09ED  2309 ;  INPUTS:  R5  NET UCB address.
                     09ED  2310 ;           R4  Scratch
                     09ED  2311 ;           R3  Local link number.
                     09ED  2312 ;           R2  Disconnect reason code
                     09ED  2313 ;           R1  Logical-link's remote node address
                     09ED  2314 ;-
                     09ED  2315 DISC_ONE:                                 ; Disconnect single logical-link
          02A7 30    09ED  2316         BSBW    XWB_LOCLNK                ; Find the logical-link XWB
     50     14 3C      09F0  2317         MOVZWL  S^#SS$_BADPARAM,R0        ; Assume no such link exists
       11 55  E8    09F3  2318         BLBS    R5,120$                   ; If LBS then XWB was not found
         3A A5  B5    09F6  2319         TSTW    XWB$W_REMNOD(R5)          ; Remote node 0?
               06 13      09F9  2320         BEQL    100$                      ; If so, ignore node check
3A A5     51  B1      09FB  2321         CMPW    R1,XWB$W_REMNOD(R5)       ; Same remote node ?
```

```
                06  12  09FF  2322                BNEQ    120$                    ; If not, return error
             0019   30  0A01  2323  100$:         BSBW    180$                    ; Disconnect the link
          50   01   D0  0A04  2324                MOVL    S^#SS$_NORMAL,R0        ; Success
                    05  0A07  2325  120$:         RSB
                        0A08  2326
                        0A08  2327  ;+
                        0A08  2328  ;  ABORT_ALL -  Abort all logical-links
                        0A08  2329  ;
                        0A08  2330  ;  INPUTS:  R5  NET UCB address
                        0A08  2331  ;           R4  Scratch
                        0A08  2332  ;           R3  Scratch
                        0A08  2333  ;           R2  Scratch
                        0A08  2334  ;           R1  Ptr to LTB
                        0A08  2335  ;
                        0A08  2336  ;-
                        0A08  2337  ABORT_ALL:                                  ; Abort all logical-links
55   E0 A1   DE         0A08  2338                MOVAL   -XWB$L_LINK -           ;
                        0A0C  2339                        +LTB$L_XWB(R1),R5       ; Prepare for scan
55   2C A5   D0         0A0C  2340  140$:         MOVL    XWB$L_LINK(R5),R5       ; Get next XWB
                07  13  0A10  2341                BEQL    160$                    ; If NEQ then got one
          52    08  B0  0A12  2342                MOVW    #NET$C_DR_THIRD,R2      ; Reason is "third party abort"
                06  10  0A15  2343                BSBB    180$                    ; Mark link to be broken
                F3  11  0A17  2344                BRB     140$                    ; Loop
          50    01  D0  0A19  2345  160$:         MOVL    S^#SS$_NORMAL,R0        ; Success
                    05  0A1C  2346                RSB
                        0A1D  2347
                        0A1D  2348
                        0A1D  2349  ;
                        0A1D  2350  ;  Disconnect the link
                        0A1D  2351  ;
   46 A5        B1      0A1D  2352  180$:         CMPW    XWB$W_X_REASON(R5),-    ; Remote reason been setup yet?
0064 8F                 0A20  2353                        #NET$C_DR_INVALID       ;
                04  12  0A23  2354                BNEQ    190$                    ; If NEQ then yes
46 A5    52     B0      0A25  2355                MOVW    R2,XWB$W_X_REASON(R5)   ; Enter disconnect reason
   44 A5        B1      0A29  2356  190$:         CMPW    XWB$W_R_REASON(R5),-    ; Local reason been setup yet?
0064 8F                 0A2C  2357                        #NET$C_DR_INVALID       ;
                04  12  0A2F  2358                BNEQ    195$                    ; If NEQ then yes
44 A5    52     B0      0A31  2359                MOVW    R2,XWB$W_R_REASON(R5)   ;
     F5C8'      30      0A35  2360  195$:         BSBW    NET$MARK_LINK           ; Mark the link to be broken
   50    6A     7D      0A38  2361                MOVQ    (R10),R0                ; Restore R0,R1,R2
52   08 AA      D0      0A3B  2362                MOVL    8(R10),R2               ;
                    05  0A3F  2363                RSB                             ; Done
                        0A40  2364  ;
                        0A40  2365  ;  Find XWB, verify access rights by PID
                        0A40  2366  ;
             0254   30  0A40  2367  200$:         BSBW    XWB_LOCLNK              ; Find XWB via local link number
                01  95  0A43  2368                TSTB    #1                      ; Clear Z-bit, assuming error
          0A 55    E8  0A45  2369                BLBS    R5,220$                 ; If LBS then no XWB
34 A5     50    D1      0A48  2370  210$:         CMPL    R0,XWB$L_PID(R5)        ; Is the process the owner ?
                04  12  0A4C  2371                BNEQ    220$                    ; If NEQ then no
                03  91  0A4E  2372                CMPB    #XWB$C_STA_CIR,-        ;
             1E A5      0A50  2373                        XWB$B_STA(R5)           ; Verify state
                    05  0A52  2374  220$:         RSB
                        0A53  2375
                        0A53  2376
                        0A53  2377  ;+
                        0A53  2378  ;  BRDCST   -   Broadcast a mailbox message
```

```
                                0A53  2379 ;
                                0A53  2380 ;  INPUTS:  R5  NET UCB address
                                0A53  2381 ;           R4  Ptr to mailbox msg text
                                0A53  2382 ;           R3  Associated mailbox mask (0 if broadcast to all mailboxes)
                                0A53  2383 ;           R2  Mailbox msg code
                                0A53  2384 ;           R1  Scratch
                                0A53  2385 ;
                                0A53  2386 ;-
                                0A53  2387 BRDCST:                                      ; Broadcast mailbox message
                                0A53  2388         ;
                                0A53  2389         ; & Code to set up R3 here will move to NETACP, eventually
                                0A53  2390         ;
      58   F791  CF    9E       0A53  2391         MOVAB   MBX_TABLE,R8                 ; Point to filter mapping table
           53    88    D0       0A58  2392 300$:   MOVL    (R8)+,R3                    ; Get next mask
                 05    13       0A5B  2393         BEQL    320$                        ; If EQL at end of table - take the msg
           52    88    B1       0A5D  2394         CMPW    (R8)+,R2                    ; Is this the msg being sent?
                 F6    12       0A60  2395         BNEQ    300$                        ; If NEQ no - loop; else, R3 has bit
           58    52    D0       0A62  2396 320$:   MOVL    R2,R8                       ; Transfer msg type code
                 00    DD       0A65  2397         PUSHL   #0                          ; Assume no message text
           57    5E    D0       0A67  2398         MOVL    SP,R7                       ; Point to it
                 54    D5       0A6A  2399         TSTL    R4                          ; Any message text?
                 26    13       0A6C  2400         BEQL    400$                        ; If EQL no, goto end of loop
           52    64    9A       0A6E  2401         MOVZBL  (R4),R2                     ; Get count field value
                 52    D6       0A71  2402         INCL    R2                          ; Inc to get total string size
           57    54    D0       0A73  2403         MOVL    R4,R7                       ; Setup stable string pointer
                 1C    11       0A76  2404         BRB     400$                        ; Jump to end of loop
                               0A78  2405
                 53    D5       0A78  2406 340$:   TSTL    R3                          ; Will everyone take this message?
                 06    13       0A7A  2407         BEQL    360$                        ; If EQL yes
      44 A5      53    D3       0A7C  2408         BITL    R3,UCB$L_DEVDEPEND(R5)      ; Can this UCB take this message?
                 12    13       0A80  2409         BEQL    400$                        ; If EQL no - don't even try to send
                 2C    BB       0A82  2410 360$:   PUSHR   #^M<R2,R3,R5>               ; Save regs
               00EA    30       0A84  2411         BSBW    NET$SEND_MBX                ; Call co-routine to setup the message
            08 50      E9       0A87  2412         BLBC    R0,380$                     ; If LBC then error
         51    57      D0       0A8A  2413         MOVL    R7,R1                       ; Get message pointer
               02DB    30       0A8D  2414         BSBW    NET$MOV_CSTR                ; Move the string with count field
                 9E    16       0A90  2415         JSB     @(SP)+                      ; Complete the message
                 2C    BA       0A92  2416 380$:   POPR    #^M<R2,R3,R5>               ; Recover regs
      55    30 A5    D0       0A94  2417 400$:   MOVL    UCB$L_LINK(R5),R5           ; Get next UCB
                 DE    12       0A98  2418         BNEQ    340$                        ; If NEQ then got one
                 8E    D5       0A9A  2419         TSTL    (SP)+                       ; Fix the stack
         50      01    D0       0A9C  2420         MOVL    S^#SS$_NORMAL,R0            ; Exit with success
                       05       0A9F  2421         RSB
                               0AA0  2422
                               0AA0  2423
                               0AA0  2424 ;+
                               0AA0  2425 ;  REPLY     -   Send general message to assocaited mailbox
                               0AA0  2426 ;
                               0AA0  2427 ;  INPUTS:  R5  NET UCB address
                               0AA0  2428 ;           R4  Ptr to mailbox msg text
                               0AA0  2429 ;           R3  & Associated mailbox mask if NETUPD$_BRDCST (0 if broadcast all)
                               0AA0  2430 ;           R2  Mailbox msg code
                               0AA0  2431 ;           R1  Scratch
                               0AA0  2432 ;
                               0AA0  2433 ;-
      58    52    D0       0AA0  2434 REPLY:  MOVL    R2,R8                       ; Get mailbox message code
                 13    11       0AA3  2435         BRB     500$                        ; Continue in common
```

```
                        0AA5  2436
                        0AA5  2437 ;+
                        0AA5  2438 ;  DECLARE  -   Pass NCB to Declared-name mailbox
                        0AA5  2439 ;
                        0AA5  2440 ;  INPUTS:  R5  NET UCB address
                        0AA5  2441 ;           R4  Ptr to NCB counted string
                        0AA5  2442 ;           R3  Scratch
                        0AA5  2443 ;           R2  Scratch
                        0AA5  2444 ;           R1  Scracth
                        0AA5  2445 ;
                        0AA5  2446 ;-
                        0AA5  2447 DECLARE:                                     ; Pass NCB to declare-object mailbox
      01EF   30         0AA5  2448          BSBW    XWB_LOCLNK                ; Find link's XWB
   29 55     E8         0AA8  2449          BLBS    R5,560$                   ; Br if no XWB
      03     91         0AAB  2450          CMPB    #XWB$C_STA_CIR,-          ; Must be in CIR state
   1E A5                0AAD  2451                  XWB$B_STA(R5)             ;
      23     12         0AAF  2452          BNEQ    560$                      ; If not then cannot redirect connect
34 A5    51   D0         0AB1  2453          MOVL    R1,XWB$L_PID(R5)          ; Set PID of process
   58    32   3C         0AB5  2454          MOVZWL  #MSG$_CONNECT,R8          ; Setup mailbox message type
54    10 AA   7D         0AB8  2455 500$:    MOVQ    R4_OFF(R10),R4            ; Get mbx message and UCB addresses
   52    54   D0         0ABC  2456          MOVL    R4,R2                     ; Copy msg pointer
         03   13         0ABF  2457          BEQL    520$                      ; If EQL then no text
   52    62   9A         0AC1  2458          MOVZBL  (R2),R2                   ; Set count of bytes to be sent
      00AA   30         0AC4  2459 520$:    BSBW    NET$SEND_MBX              ; Prepare to send mailbox message
      0D 50  E9         0AC7  2460          BLBC    R0,580$                   ; Br on error
   51    54   D0         0ACA  2461          MOVL    R4,R1                     ; Copy NCB pointer
         03   13         0ACD  2462          BEQL    540$                      ; Skip if null
      0299   30         0ACF  2463          BSBW    NET$MOV_CSTR              ; Move counted string into buffer
         9E   16         0AD2  2464 540$:    JSB     @(SP)+                    ; Complete writing mailbox
   50    01   D0         0AD4  2465 560$:    MOVL    S^#SS$_NORMAL,R0          ; Success
               05         0AD7  2466 580$:    RSB                               ; Done
                          0AD8  2467
                          0AD8  2468
                          0AD8  2469 ;+
                          0AD8  2470 ;  DLLTRN   -   Datalink state transition
                          0AD8  2471 ;
                          0AD8  2472 ;  INPUTS:  R5  NET UCB address
                          0AD8  2473 ;           R4  Scratch
                          0AD8  2474 ;           R3  Scratch
                          0AD8  2475 ;           R2  Scratch
                          0AD8  2476 ;           R1  Ptr to datalink's LPD
                          0AD8  2477 ;
                          0AD8  2478 ;-
52    34 A5    D0         0AD8  2479 DLLTRN: MOVL    UCB$L_VCB(R5),R2          ; Get RCB
         01    91         0ADC  2480          CMPB    #LPD$C_LOC_INX,-          ; Is this the local LPD
      20 A1                0ADE  2481           LPD$B_PTH_INX(R1)                ;
         4B    12         0AE0  2482          BNEQ     UNKNOWN                  ; If not, branch
                          0AE2  2483                                            ;
         3F    BB         0AE2  2484          PUSHR   #^M<R0,R1,R2,R3,R4,R5>    ; Save regs
   53    55   D0         0AE4  2485          MOVL    R5,R3                     ; Copy UCB address
55    30 A2   D0         0AE7  2486          MOVL    RCB$L_PTR_TQE(R2),R5      ; Get TQE
         02    E0         0AEB  2487          BBS     #TQE$V_REPEAT,-           ; Br if timer is in use
   3B 0B A5                0AED  2488                  TQE$B_RQTYPE(R5),600$     ;
         05    90         0AF0  2489          MOVB    #TQE$C_SSREPT,-           ; Set for system subroutine repeat
      0B A5                0AF2  2490                  TQE$B_RQTYPE(R5)          ;
   0000'CF    9E         0AF4  2491          MOVAB   W^NET$TIMER,-             ; Set timer handler address
      0C A5                0AF8  2492                  TQE$L_FPC(R5)             ;
```

```
                    14 A5    53   D0  0AFA  2493          MOVL    R3,TQE$L_FR4(R5)        ; Save UCB address
00000000 00989680 8F   7D  0AFE  2494          MOVQ    #10*1000*1000,-         ; 1 tick = 1 sec
                    20 A5        0B08  2495                  TQE$Q_DELTA(R5)         ;
              50    F60A CF    9E  0B0A  2496          MOVAB   W^NET$GL_OFF_DPTFLG,R0  ; Get address of offset to DPT$B_FLAGS
                    50   60    C0  0B0F  2497          ADDL    (R0),R0                 ; Make it an address
                    60   04    88  0B12  2498          BISB    #DPT$M_NOUNLOAD,(R0)    ; Prevent reload of driver
           50    00000000'GF   7D  0B15  2499          MOVQ    G^EXE$GQ_SYSTIME,R0     ; Set time of first tick
                               0B1C  2500          DSBINT  #IPL$_TIMER             ; Lower IPL to that of timer service
                 00000000'GF   16  0B22  2501          JSB     G^EXE$INSTIMQ           ; Insert into queue
                               0B28  2502          ENBINT                  ; Restore IPL
                         3F    BA  0B2B  2503 600$:   POPR    #^M<R0,R1,R2,R3,R4,R5>  ;
                               0B2D  2504                                  ;
                               0B2D  2505 UNKNOWN:                         ;
                       F4D0'   30  0B2D  2506          BSBW    TR$UPDATE               ; Fct code is in R0
                               05  0B30  2507          RSB                             ; Exit with status in R0
                               0B31  2508
                               0B31  2509                  .DSABL  LSB
                               0B31  2510
```

```
                     0B31  2512 .SBTTL  NET$SEND_CS_MBX - Send counted string to mailbox
                     0B31  2513 ;+
                     0B31  2514 ;
                     0B31  2515 ;  A mailbox message is built and sent to the mailbox associated with the UCB
                     0B31  2516 ;  associated with the XWB.  The counted string pointed to by R1 is appended to
                     0B31  2517 ;  the end of the mailbox message.  R2 contains the assumed total count of the
                     0B31  2518 ;  string and may be zero.  If there is no mailbox then the routine is assumed
                     0B31  2519 ;  implicitly successful.
                     0B31  2520 ;
                     0B31  2521 ;
                     0B31  2522 ;  INPUTS:  R8  Mailbox message type code
                     0B31  2523 ;           R5  XWB address
                     0B31  2524 ;           R2  Assumed total length of string (low byte only)
                     0B31  2525 ;           R1  Address of count field of string
                     0B31  2526 ;
                     0B31  2527 ;  OUTPUTS: R2  Zero
                     0B31  2528 ;           R1  Garbage
                     0B31  2529 ;           R0  SS$_NORMAL      if mailbox successfully written
                     0B31  2530 ;               SS$_NOMBX!1     if no associated mailbox or no UCB
                     0B31  2531 ;               Zero            if (R1)+1 NEQ R2  or  R2 GTRU 17
                     0B31  2532 ;               Also see        NET$SEND_MBX for R0 error codes
                     0B31  2533 ;
                     0B31  2534 ;
                     0B31  2535 ;       All other registers are preserved
                     0B31  2536 ;
                     0B31  2537 ;-
                     0B31  2538 NET$SEND_CS_MBX::
           3E   BB   0B31  2539          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save regs
                     0B33  2540                                          ;
           52   D5   0B33  2541          TSTL    R2                      ; Any bytes in string ?
           06   12   0B35  2542          BNEQ    10$                     ; If NEQ yes, else can't trust R1
       6E  70'AF   9E   0B37  2543          MOVAB   B^50$,(SP)              ; Setup null string ptr
           0F   11   0B3B  2544          BRB     20$                     ; Continue
           50   D4   0B3D  2545 10$:     CLRL    R0                      ; Assume string error
       11   52   D1   0B3F  2546          CMPL    R2,#17                  ; Is count within range
           27   1A   0B42  2547          BGTRU   40$                     ; If not, branch
    53     52   61   83   0B44  2548          SUBB3   (R1),R2,R3              ; Check count field consistency
           53   97   0B48  2549          DECB    R3                      ; Account for count field itself
           1F   12   0B4A  2550          BNEQ    40$                     ; Inconsistent if NEQ
    50   0275   8F   3C   0B4C  2551 20$:     MOVZWL  #SS$_NOMBX!1,R0         ; Assume no UCB or mailbox
    55     10   A5   D0   0B51  2552          MOVL    XWB$L_ORGUCB(R5),R5     ; Get UCB
           14   13   0B55  2553          BEQL    40$                     ; If none, done
         60   A5   D5   0B57  2554          TSTL    UCB$L_AMB(R5)           ; Is there a mailbox ?
           0F   13   0B5A  2555          BEQL    40$                     ; If not, branch
         0012   30   0B5C  2556          BSBW    NET$SEND_MBX            ; Build header (co-routine)
       09 50   E9   0B5F  2557          BLBC    R0,40$                  ; Br on error
    51     04   AE   D0   0B62  2558          MOVL    4(SP),R1                ; Get string address (note stack)
         0202   30   0B66  2559          BSBW    NET$MOV_CSTR            ; Move string with count field
           9E   16   0B69  2560          JSB     @(SP)+                  ; Close and send mbx message
                     0B6B  2561                                          ;
           3E   BA   0B6B  2562 40$:     POPR    #^M<R1,R2,R3,R4,R5>     ; Recover regs
           52   D4   0B6D  2563          CLRL    R2                      ; String has been consumed
                05   0B6F  2564          RSB
                     0B70  2565
                00   0B70  2566 50$:     .BYTE   0                       ; Phony counted string for mailbox
```

```
                        0B71  2568 .SBTTL  NET$SEND_MBX    - Co-routine to send mailbox message
                        0B71  2569 ;+
                        0B71  2570 ;
                        0B71  2571 ;  The first time the routine is entered the associated mailbox is found, a
                        0B71  2572 ;  buffer is allocated for the message, and the mailbox header is built.  When
                        0B71  2573 ;  the routine is re-entered, after a call-back to the co-routine, the message
                        0B71  2574 ;  is closed and sent to the mailbox.
                        0B71  2575 ;
                        0B71  2576 ;  The original entry parameters are given below, the re-entry parameters are
                        0B71  2577 ;  given within the body of the code.
                        0B71  2578 ;
                        0B71  2579 ;
                        0B71  2580 ;  INPUTS:  R8  Mailbox message type code
                        0B71  2581 ;           R5  UCB address
                        0B71  2582 ;           R3  Scratch
                        0B71  2583 ;           R2  Count of bytes co-routine will enter into message
                        0B71  2584 ;           R1  Scratch
                        0B71  2585 ;           R0  Scratch
                        0B71  2586 ;
                        0B71  2587 ;  OUTPUTS: R3  Pointer to next byte in mailbox message to be filled
                        0B71  2588 ;           R2  Address of allocated buffer if R0=SS$_NORMAL
                        0B71  2589 ;           R1  Garbage
                        0B71  2590 ;           R0  SS$_NORMAL  if successful
                        0B71  2591 ;               SS$_NOMBX   if there's no associated mailbox
                        0B71  2592 ;
                        0B71  2593 ;               see NET$ALONONPAGED for additional error status
                        0B71  2594 ;
                        0B71  2595 ;           All other registers are preserved
                        0B71  2596 ;
                        0B71  2597 ;-
                        0B71  2598 NET$SEND_MBX::
                        0B71  2599          ;
                        0B71  2600          ;
                        0B71  2601          ;    Add 24 to the number of bytes the user will enter.  This will
                        0B71  2602          ;    ensure that the allocated block is large enough for COM$DRVDEALMEM
                        0B71  2603          ;    to deallocate -- also creates space for:
                        0B71  2604          ;
                        0B71  2605          ;       12 bytes for  standard buffer header
                        0B71  2606          ;        2 bytes for mailbox msg type code
                        0B71  2607          ;        2 bytes for mailbox  unit number
                        0B71  2608          ;        1 byte  for count field for device name
                        0B71  2609          ;
                        0B71  2610          ;
    52    18    C0      0B71  2611          ADDL    #24,R2                  ; Increase buffer size
50    28 A5    D0      0B74  2612          MOVL    UCB$L_DDB(R5),R0        ; DDB pointer
51    14 A0    9E      0B78  2613          MOVAB   DDB$T_NAME(R0),R1       ; Get device name string ptr
                        0B7C  2614                                          ;
          51    DD      0B7C  2615          PUSHL   R1                      ; Save device name string ptr
    51    61    9A      0B7E  2616          MOVZBL  (R1),R1                 ; Get string size
    51    52    C0      0B81  2617          ADDL    R2,R1                   ; Add in remaining bytes
       01AD     30      0B84  2618          BSBW    NET$ALONONPAGED         ; Get the buffer
          51 8ED0      0B87  2619          POPL    R1                      ; Restore device name string ptr
                        0B8A  2620                                          ;
       01 50    E8      0B8A  2621          BLBS    R0,10$                  ; If LBS then okay
                05      0B8D  2622          RSB                             ; Return with error status in R0
                        0B8E  2623
53    52    0C    C1    0B8E  2624 10$:     ADDL3   #12,R2,R3               ; Get pointer to start of msg
```

```
                83   58    B0   0B92   2625          MOVW    R8,(R3)+                ; Enter message type code
             83   54 A5    B0   0B95   2626          MOVW    UCB$W_UNIT(R5),(R3)+    ; Enter unit I.D.
                01CF       30   0B99   2627          BSBW    NET$MOV_CSTR            ; Move in device name with count field
                50   01    D0   0B9C   2628          MOVL    S^#SS$_NORMAL,R0        ; Indicate success
                     9E    16   0B9F   2629          JSB     @(SP)+                  ; Call co-routine for more bytes
                                0BA1   2630                                          ; Note that R4 is unmodified
                                0BA1   2631          ;
                                0BA1   2632          ;
                                0BA1   2633          ;    On coroutine return:       R5 = UCB address
                                0BA1   2634          ;                               R3 = address of 1st byte past mbx msg
                                0BA1   2635          ;                               R2 = buffer address
                                0BA1   2636          ;
                                0BA1   2637          ;    On return to caller:       R0 = EXE$WRITEMBX status
                                0BA1   2638          ;                               R1-R5 are garbage
                                0BA1   2639          ;
                                0BA1   2640          ;
          54    52   0C    C1   0BA1   2641          ADDL3   #12,R2,R4               ; Get start of mbx message
                53   54    C2   0BA5   2642          SUBL    R4,R3                   ; Get length of mbx message
          50    0274 8F    3C   0BA8   2643          MOVZWL  #SS$_NOMBX,R0           ; Assume no mailbox
             55   60 A5    D0   0BAD   2644          MOVL    UCB$L_AMB(R5),R5        ; Get mailbox
                   06    13   0BB1   2645          BEQL    20$                     ; If EQL then no mailbox
           00000000'GF     16   0BB3   2646          JSB     G^EXE$WRTMAILBOX        ; Send message to mailbox
                                0BB9   2647                                          ;
                   50    DD   0BB9   2648 20$:     PUSHL   R0                      ; Save return status
          50   54   0C    C3   0BBB   2649          SUBL3   #12,R4,R0               ; Get buffer address
                018C       30   0BBF   2650          BSBW    NET$DEALLOCATE          ; Deallocate block in R0
                   50 8ED0    0BC2   2651          POPL    R0                      ; Restore reg
                              0BC5   2652                                          ;
                   05         0BC5   2653          RSB                             ; Done
                              0BC6   2654
```

```
                     0BC6  2656 .SBTTL  NET$CREATE_XWB  - Create XWB for logical-link
                     0BC6  2657 ;++
                     0BC6  2658 ;
                     0BC6  2659 ;  An XWB (the logical-link control structure that will eventually be attached
                     0BC6  2660 ;  to an I/O channel CCB$L_WIND field) is allocated and initialized, provided
                     0BC6  2661 ;  that the current maximum logical-link count is not exceeded.  The current
                     0BC6  2662 ;  logical-link count in incremented.
                     0BC6  2663 ;
                     0BC6  2664 ;  No local link address is assigned, and the XWB is not linked into the LTB.
                     0BC6  2665 ;
                     0BC6  2666 ;
                     0BC6  2667 ;  INPUTS:       R5      NET UCB Address
                     0BC6  2668 ;                R3      Remote node address
                     0BC6  2669 ;                R0      Scratch
                     0BC6  2670 ;
                     0BC6  2671 ;  OUTPUTS:      R5      Address of XWB if successful, otherwise LBS
                     0BC6  2672 ;                R0      Status
                     0BC6  2673 ;
                     0BC6  2674 ;                All other registers are preserved
                     0BC6  2675 ;
                     0BC6  2676 ;--
                     0BC6  2677 NET$CREATE_XWB::                                     ; Get idle XWB
             1E   BB 0BC6  2678         PUSHR  #^M<R1,R2,R3,R4>                     ; Save regs to be used
                     0BC8  2679         ;
                     0BC8  2680         ;
                     0BC8  2681         ;    Make sure we are not over our limit (MCOUNT = current links + 1).
                     0BC8  2682         ;
                     0BC8  2683         ;
         50   027C 8F   3C  0BC8  2684         MOVZWL #SS$_NOLINKS,R0                  ; Assume failure
           52   34 A5   D0  0BCD  2685         MOVL    UCB$L_VCB(R5),R2                ; Point to RCB
                30 13   13  0BD1  2686         BEQL    13$                             ; If EQL then no RCB
           51   54 A2   3C  0BD3  2687         MOVZWL  RCB$W_MCOUNT(R2),R1             ; Get current Mount Count
                2A 13   13  0BD7  2688         BEQL    13$                             ; If EQL, NETACP shutting down
           58 A2   51   B1  0BD9  2689         CMPW    R1,RCB$W_MAX_LNK(R2)            ; Is new link allowed ?
                19 1A   1A  0BDD  2690         BGTRU   12$                             ; If not, branch
         51   017C 8F   3C  0BDF  2691         MOVZWL #XWB_C_LEN,R1                    ; Get size of XWB
                013C   30  0BE4  2692         BSBW    NET$ACONPGD_Z                   ; Allocate the block and zero it
                       0BE7  2693                                                     ; to initialize most fields
           51   52   D0  0BE7  2694         MOVL    R2,R1                           ; Save XWB pointer
           52   34 A5   D0  0BEA  2695         MOVL    UCB$L_VCB(R5),R2                ; Point to RCB
           55   51   D0  0BEE  2696         MOVL    R1,R5                           ; Use standard XWB pointer
           54   08 AE   B0  0BF1  2697         MOVW    8(SP),R4                        ; Get dst node address
                10 50   E8  0BF5  2698         BLBS    R0,15$                          ; Br if successful
                       0BF8  2699 12$:   BUMP    W,RCB$W_CNT_XRE(R2)             ; Account for resource error
              55   01   D0  0C03  2700 13$:   MOVL    #1,R5                           ; Invalidate XWB ptr
                14   11  0C06  2701         BRB     100$                            ; Done
                       0C08  2702 15$:   ;
                       0C08  2703        ;
                       0C08  2704        ;    Initialize the XWB and bump RCB mount count.
                       0C08  2705        ;
                       0C08  2706        ;
                15   10  0C08  2707         BSBB    INIT_XWB                        ; Init XWB
 009E C2       54 A2   B1  0C0A  2708         CMPW    RCB$W_MCOUNT(R2),RCB$W_CNT_MLL(R2) ; New max active links value?
                04   1B  0C10  2709         BLEQU   30$                             ; If LEQU then no
              009E C2   B6  0C12  2710         INCW    RCB$W_CNT_MLL(R2)               ; Bump max active link count
                       0C16  2711                                                     ; (#links = MCOUNT-1)
                54 A2   B6  0C16  2712 30$:   INCW    RCB$W_MCOUNT(R2)                ; Account for new link
```

```
                50   01    D0   0C19  2713          MOVL    #1,R0                           ; Success
                     1E    BA   0C1C  2714 100$:     POPR    #^M<R1,R2,R3,R4>                ; Restore regs
                     05    0C1E  2715                RSB                                     ; Done
                           0C1F  2716
                           0C1F  2717 INIT_XWB:                                              ; Initialize XWB.
              1F A5    08   90   0C1F  2718          MOVB    #NET$C_IPL,       XWB$B_FIPL(R5)      ; Setup fork IPL
              0A A5    1C   90   0C23  2719          MOVB    #DYN$C_NDB,       XWB$B_TYPE(R5)      ; Setup structure type
              1E A5    00   90   0C27  2720          MOVB    #XWB$C_STA_CLO,   XWB$B_STA(R5)       ; Init logical-link state
              0E A5    10   B0   0C2B  2721          MOVW    #XWB$M_STS_CON,   XWB$W_STS(R5)       ; Init the status word
          1C A5    0200 8F   B0   0C2F  2722          MOVW    #XWB$M_FLG_CLO,   XWB$W_FLG(R5)       ; Init FLG bits
          44 A5    0064 8F   B0   0C35  2723          MOVW    #NET$C_DR_INVALID,XWB$W_R_REASON(R5)  ; Init rcv'd discon reason
          46 A5    0064 8F   B0   0C3B  2724          MOVW    #NET$C_DR_INVALID,XWB$W_X_REASON(R5)  ; Init xmt'd discon reason
             3A A5    54   B0   0C41  2725          MOVW    R4,               XWB$W_REMNOD(R5)    ; Setup remote node i.d.
             30 A5    52   D0   0C45  2726          MOVL    R2,               XWB$L_VCB(R5)       ; Setup VCB address
          016B C5     80 8F   90   0C49  2727          MOVB    #^X<80>,          XWB$$+ACB$B_RMOD(R5); Setup Special Kernal AST
      0178 C5    00000000'EF   9E   0C4F  2728          MOVAB   NET$KAST,         XWB$$+ACB$L_KAST(R5); mode and address
                           0C58  2729                                                        ;
                50   0118 C5   9E   0C58  2730          MOVAB   XWB$Q_FREE_CXB(R5),R0             ; Get free queue address
                     60   50   D0   0C5D  2731          MOVL    R0,(R0)                           ; Init queue header
                     60   80   DE   0C60  2732          MOVAL   (R0)+,(R0)                        ;
                           0C63  2733                                                        ;
             54 A5    63 A2   9B   0C63  2734          MOVZBW  RCB$B_ECL_RFA(R2),  XWB$W_RETRAN(R5)  ; Set default rexmit's
             56 A5    64 A2   9B   0C68  2735          MOVZBW  RCB$B_ECL_DFA(R2),  XWB$W_DLY_FACT(R5); Set default delay factor
             58 A5    65 A2   9B   0C6D  2736          MOVZBW  RCB$B_ECL_DWE(R2),  XWB$W_DLY_WGHT(R5); Set default delay weight
             42 A5    7C A2   B0   0C72  2737          MOVW    RCB$W_ECLSEGSIZ(R2),XWB$W_REMSIZ(R5)  ; Set temp 'seg' size
          50 A5    76 A2    01   A1   0C77  2738          ADDW3   #1,RCB$W_TIM_CNI(R2),XWB$W_TIMER(R5)  ; Set inbound connect timer
                           0C7D  2739                                                        ; (#1 is for clock skew)
                           0C7D  2740          ;
                           0C7D  2741          ;
                           0C7D  2742          ;   Build the route header.
                           0C7D  2743          ;
                           0C7D  2744          ;
    51    015E C5    9E   0C7D  2745          MOVAB   XWB$T_TR3HDR+6(R5),R1     ; Setup route-header pointer
                     71   94   0C82  2746          CLRB    -(R1)                     ; Zero the "visits" field
       71    0E A2    B0   0C84  2747          MOVW    RCB$W_ADDR(R2),-(R1)      ; Enter src node address
             71   54   B0   0C88  2748          MOVW    R4,-(R1)                  ; Enter dst node address
             71   02   90   0C8B  2749          MOVB    #TR3$C_MSG_DATA,-(R1)     ; Enter message type
    0120 C5    51   D0   0C8E  2750          MOVL    R1,XWB$L_PTR_RTHD(R5)     ; Setup route-header pointer
             71   06   D0   0C93  2751          MOVL    #6,-(R1)                  ; Store the route-header size
                     05   0C96  2752          RSB                               ; Done
                          0C97  2753
```

```
                     0C97  2755 .SBTTL  XWB_LOCLNK      - Get XWB via local link number
                     0C97  2756 ;+
                     0C97  2757 ;
                     0C97  2758 ;  INPUTS:      R5      Any NET UCB address
                     0C97  2759 ;               R3      Local link number
                     0C97  2760 ;
                     0C97  2761 ;  OUTPUTS:     R5      Address of associated XWB, or low bit set if none
                     0C97  2762 ;
                     0C97  2763 ;               All other registers are preserved.
                     0C97  2764 ;-
                     0C97  2765 XWB_LOCLNK:                                  ; Get XWB context
           14   BB  0C97  2766          PUSHR   #^M<R2,R4>                  ; Save reg
                0C99  2767                                                  ;
 52  34 A5   D0  0C99  2768          MOVL    UCB$L_VCB(R5),R2               ; Get RCB address
           05   12  0C9D  2769          BNEQ    5$                          ; If NEQ the RCB exists
     55    01   D0  0C9F  2770          MOVL    #1,R5                       ; Invalidate XWB address
           02   11  0CA2  2771          BRB     10$                         ; Done
           03   10  0CA4  2772 5$:      BSBB    NET$XWB_LOCLNK              ; If NEQ Locate the link
                0CA6  2773                                                  ;
           14   BA  0CA6  2774 10$:     POPR    #^M<R2,R4>                  ; Restore reg
           05   0CA8  2775          RSB
                0CA9  2776
                0CA9  2777
                0CA9  2778 .SBTTL  NET$XWB_LOCLNK  - Get XWB via local link number
                0CA9  2779 ;++
                0CA9  2780 ;
                0CA9  2781 ;  The Link Table is located and the slot associated with the specified link
                0CA9  2782 ;  number is found.  If this slot contains an XWB then the link sequence number
                0CA9  2783 ;  is checked.  If there is a sequence number mismatch, or if there is no
                0CA9  2784 ;  active XWB, then the low bit of R5 is set.  Else, the XWB address is stored
                0CA9  2785 ;  in R5.
                0CA9  2786 ;
                0CA9  2787 ;
                0CA9  2788 ;  INPUTS:      R5,R4   Scratch
                0CA9  2789 ;               R3      Local link number - high order word is clear
                0CA9  2790 ;               R2      RCB address
                0CA9  2791 ;
                0CA9  2792 ;  OUTPUTS:     R5      Address of associated XWB, or low bit set if none
                0CA9  2793 ;               R4      LTB (link table) address
                0CA9  2794 ;
                0CA9  2795 ;               All other registers are preserved.
                0CA9  2796 ;
                0CA9  2797 ;--
                0CA9  2798 NET$XWB_LOCLNK::                                 ; Locate XWB via local link number
      54   24 A2   D0  0CA9  2799          MOVL    RCB$L_PTR_LTB(R2),R4       ; Get Link Table pointer
           1E   13  0CAD  2800          BEQL    20$                         ; Return error if not there
55    53  FFFFFC00 8F   CB  0CAF  2801          BICL3   #^C<NET$C_MAXLNK>,R3,R5 ; Get link 'index'
           14   13  0CB7  2802          BEQL    20$                         ; Index '0' isn't used
      04 A4   55   B1  0CB9  2803          CMPW    R5,LTB$W_SLT_TOT(R4)        ; Index within range ?
           0E   1A  0CBD  2804          BGTRU   20$                         ; If not, branch
   55   10 A445   D0  0CBF  2805          MOVL    LTB$L_SLOTS(R4)[R5],R5      ; Get XWB address
        09   55   E8  0CC4  2806          BLBS    R5,30$                      ; If LBS then none
   3E A5    53   B1  0CC7  2807          CMPW    R3,XWB$W_LOCLNK(R5)         ; Sequence number match ?
           03   13  0CCB  2808          BEQL    30$                         ; If so, branch
      55   01   88  0CCD  2809 20$:     BISB    #1,R5                       ; Flag no associated XWB
                05  0CD0  2810 30$:     RSB
```

```
                                0CD1  2812 .SBTTL  NET$RET_SLOT    - Return logical-link XWB slot if done
                                0CD1  2813 .SBTTL  NET$QUE_XWB     - Queue XWB to NETACP's AQB
                                0CD1  2814 ;++
                                0CD1  2815 ;
                                0CD1  2816 ;  If the XWB is busy then the queue attempt is aborted.  If the XWB is
                                0CD1  2817 ;  not busy then the XWB$V_STS_SOL bit is set to prevent any further XWB use.
                                0CD1  2818 ;
                                0CD1  2819 ;
                                0CD1  2820 ;
                                0CD1  2821 ;  INPUTS:       R5       XWB pointer
                                0CD1  2822 ;
                                0CD1  2823 ;  OUTPUTS:      R0,R1    Zero
                                0CD1  2824 ;
                                0CD1  2825 ;                 All other registers are preserved.
                                0CD1  2826 ;
                                0CD1  2827 ;
                                0CD1  2828 ;--
                                0CD1  2829 NET$RET_SLOT::                                   ; Return logical-link if done
          1E A5   00    91      0CD1  2830          CMPB    #XWB$C_STA_CLO,XWB$B_STA(R5)      ; In 'closed' state?
                  06    13      0CD5  2831          BEQL    10$                               ; If so, continue
          1E A5   06    91      0CD7  2832          CMPB    #XWB$C_STA_DIR,XWB$B_STA(R5)      ; If DIR state then we've sent
                                0CDB  2833                                                    ; the DC msg already
                  11    12      0CDB  2834          BNEQ    40$                               ; If not, XWB is still active
             0C A5    B5      0CDD  2835 10$:      TSTW    XWB$W_REFCNT(R5)                  ; Any references ?
                  0C    12      0CE0  2836          BNEQ    40$                               ; If NEQ must wait
                                0CE2  2837 ;&       BBS     #XWB$V_FLG_LOCK,XWB$W_FLG(R5),40$; Exit if XWB is locked
       0E A5    0C04 8F    B3      0CE2  2838          BITW    #XWB$M_STS_ASTPND!-               ; AST pending
                                0CE8  2839                   XWB$M_STS_ASTREQ!-               ; AST requested
                                0CE8  2840                   XWB$M_STS_SOL,XWB$W_STS(R5)      ; Fork block in use
                  04    12      0CE8  2841          BNEQ    40$                               ; If NEQ, XWB is busy
                  2A    10      0CEA  2842          BSBB    NET$DRAIN_FREE_CXB                ; Drain CXB free queue
                  03    10      0CEC  2843          BSBB    NET$QUE_XWB                       ; Queue XWB to NETACP's AQB
                  50    7C      0CEE  2844 40$:      CLRQ    R0                                ; Say "nothing to xmit"
                  05    05      0CF0  2845          RSB                                       ; Done
                                0CF1  2846
                                0CF1  2847
                                0CF1  2848 NET$QUE_XWB::                                    ; Queue XWB to NETACP's AQB
                                0CF1  2849          ASSUME  IPL$_SYNCH  EQ  NET$C_IPL
                                0CF1  2850
       1F 0E A5   02    E2      0CF1  2851          BBSS    #XWB$V_STS_SOL,XWB$W_STS(R5),50$; If BS, then queue block in use
                  3C    BB      0CF6  2852          PUSHR   #^M<R2,R3,R4,R5>                  ; Save regs
                                0CF8  2853                                                    ;
          52    30 A5   D0      0CF8  2854          MOVL    XWB$L_VCB(R5),R2                  ; Get RCB
                0C A2   B6      0CFC  2855          INCW    RCB$W_TRANS(R2)                   ; Account for ACP transaction
          54    10 A2   D0      0CFF  2856          MOVL    RCB$L_AQB(R2),R4                  ; Get AQB
          04 B4     65    0E      0D03  2857          INSQUE  (R5),@AQB$L_ACPQBL(R4)            ; Queue XWB to AQB
                    0A    12      0D07  2858          BNEQ    30$                               ; If NEQ then not first
          51    0C A4   D0      0D09  2859          MOVL    AQB$L_ACPPID(R4),R1               ; Get ACP's PID
       00000000'GF      16      0D0D  2860          JSB     G^SCH$WAKE                        ; Wake the ACP
                                0D13  2861                                                    ;
                  3C    BA      0D13  2862 30$:      POPR    #^M<R2,R3,R4,R5>                  ; Restore regs
                  05    05      0D15  2863 50$:      RSB                                       ; done
                                0D16  2864
                                0D16  2865
                                0D16  2866
                                0D16  2867 .SBTTL  NET$DRAIN_FREE_CXB      - Drain CXB free queue
                                0D16  2868
```

```
                     0D16  2869 NET$DRAIN_FREE_CXB::                          ; Drain CXB free queue
                     0D16  2870          ;
                     0D16  2871          ;
                     0D16  2872          ;   All registers except for R0 must be preserved.
                     0D16  2873          ;
                     0D16  2874          ;
50   0118 D5   0F    0D16  2875 10$:     REMQUE  @XWB$Q_FREE_CXB(R5),R0          ; Get next CXB
          05   1D    0D1B  2876          BVS     20$                             ; If VS, none left
        002E   30    0D1D  2877          BSBW    NET$DEALLOCATE                  ; Deallocate block in R0
          F4   11    0D20  2878          BRB     10$                             ; Loop
               05    0D22  2879 20$:     RSB                                     ; Done
                     0D23  2880
                     0D23  2881
```

```
                                                     B 1
                     0D23  2883 .SBTTL  NET$ALONPGD_Z   - Allocate and zero from system pool
                     0D23  2884 .SBTTL  NET$ALONONPAGED - Allocate from system pool
                     0D23  2885 ;++
                     0D23  2886 ;
                     0D23  2887 ;  A buffer is allocated from non-paged pool and its size field is set to
                     0D23  2888 ;  the size requested.  Its type field is set to DYN$C_CXB.
                     0D23  2889 ;
                     0D23  2890 ;
                     0D23  2891 ;  INPUTS:  R2 = Scratch
                     0D23  2892 ;           R1 = Size, in bytes, of block to be allocated
                     0D23  2893 ;           R0 = Scratch
                     0D23  2894 ;
                     0D23  2895 ;  OUTPUTS: R2 = Address of block if successful
                     0D23  2896 ;              Zero if unsuccessful
                     0D23  2897 ;           R0 = Standard VMS status code
                     0D23  2898 ;
                     0D23  2899 ;           All other registers are preserved.
                     0D23  2900 ;
                     0D23  2901 ;--
                     0D23  2902                 .ENABL  LSB
                     0D23  2903 NET$ALONPGD_Z::                          ; Allocate and zero non-paged buffer
          0F   10    0D23  2904         BSBB    NET$ALONONPAGED          ; Allocate the buffer
        25 50  E9    0D25  2905         BLBC    R0,20$                   ; If LBC then error
                     0D28  2906                                          ;
          3F   BB    0D28  2907         PUSHR   #^M<R0,R1,R2,R3,R4,R5>   ; Save regs
62  51  00  6E  00   2C    0D2A  2908         MOVC5   #0,(SP),#0,R1,(R2)       ; Zero the entire buffer
          3F   BA    0D30  2909         POPR    #^M<R0,R1,R2,R3,R4,R5>   ; Restore regs
                     0D32  2910                                          ;
          11   11    0D32  2911         BRB     10$                      ; Setup the type and size fields (again)
                     0D34  2912
                     0D34  2913
                     0D34  2914 NET$ALONONPAGED::                        ; Allocate non-paged memory
                     0D34  2915                                          ;
          0A   BB    0D34  2916         PUSHR   #^M<R1,R3>               ; Save regs
00000000'GF    16    0D36  2917         JSB     G^EXE$ALONONPAGED        ; Allocate memory
          0A   BA    0D3C  2918         POPR    #^M<R1,R3>               ; Restore regs
                     0D3E  2919                                          ;
        04 50  E8    0D3E  2920         BLBS    R0,20$                   ; If LBS then success
          52   D4    0D41  2921         CLRL    R2                       ; Zero the buffer pointer
          08   11    0D43  2922         BRB     20$                      ; Take common exit
  08 A2   51   B0    0D45  2923 10$:    MOVW    R1,CXB$W_SIZE(R2)        ; Set size for deallocation
          1B   90    0D49  2924         MOVB    #DYN$C_CXB,-             ;
        0A A2        0D4B  2925                 CXB$B_TYPE(R2)           ; Set tentative buffer type
               05    0D4D  2926 20$:    RSB                              ; Return with status in R0
                     0D4E  2927
                     0D4E  2928                 .DSABL  LSB
                     0D4E  2929
```

```
                        0D4E  2931 .SBTTL  NET$DEALLOCATE  - Deallocate non-paged pool
                        0D4E  2932 ;+
                        0D4E  2933 ;
                        0D4E  2934 ;  IPL must be NET$C_IPL or lower.
                        0D4E  2935 ;
                        0D4E  2936 ;
                        0D4E  2937 ;  INPUTS:      R0      Address of block
                        0D4E  2938 ;
                        0D4E  2939 ;  OUTPUTS:     R0      Zero
                        0D4E  2940 ;
                        0D4E  2941 ;                       All other registers are preserved.
                        0D4E  2942 ;
                        0D4E  2943 ;-
                        0D4E  2944         ASSUME  NET$C_IPL  LE  IPL$_SYNCH       ; Can't deallocate above SYNCH
                        0D4E  2945
                        0D4E  2946 NET$DEALLOCATE::                                ; Deallocate non-paged pool
              0E   BB   0D4E  2947         PUSHR   #^M<R1,R2,R3>                   ; Save regs
              7E   D4   0D50  2948         CLRL    -(SP)                           ; Value to return in R0
                        0D52  2949                                                 ;
    51    08 A0   3C   0D52  2950         MOVZWL  8(R0),R1                        ; Get size of block
00000000'GF   16   0D56  2951         JSB     G^EXE$DEANONPGDSIZ                 ; Deallocate it
                        0D5C  2952                                                 ;
              0F   BA   0D5C  2953         POPR    #^M<R0,R1,R2,R3>                ; Restore regs
                   05   0D5E  2954         RSB                                     ; Done
                        0D5F  2955
```

```
                     0D5F  2957 .SBTTL  NET$MOV_TO_XWB  - Move counted string to XWB$B_DATA
                     0D5F  2958 .SBTTL  NET$MOV_CSTR    - Move counted string with count field
                     0D5F  2959 .SBTTL  NET$MOV_USTR    - Move counted string without count field
                     0D5F  2960 ;+
                     0D5F  2961 ;
                     0D5F  2962 ;  The source string is moved to its destination.  Both the source
                     0D5F  2963 ;  and destination pointers are updated.
                     0D5F  2964 ;
                     0D5F  2965 ;
                     0D5F  2966 ;  INPUTS:      R5      Pointer to destination field
                     0D5F  2967 ;               R1      Pointer to count field of source string
                     0D5F  2968 ;
                     0D5F  2969 ;  OUTPUTS:     R3      Pointer to first byte beyond end of destination
                     0D5F  2970 ;               R1      Pointer to first byte beyond source string
                     0D5F  2971 ;
                     0D5F  2972 ;               All other registers are preserved
                     0D5F  2973 ;
                     0D5F  2974 ;
                     0D5F  2975 ;-
                     0D5F  2976                 .ENABL  LSB
                     0D5F  2977 NET$MOV_TO_XWB::                        ; Move counted string to XWB$B_DATA
           53   DD  0D5F  2978         PUSHL   R3                       ; Save reg
53    5B A5   9E  0D61  2979         MOVAB   XWB$B_DATA(R5),R3        ; Setup destination ptr
        04   10  0D65  2980         BSBB    NET$MOV_CSTR             ; Move the string
        53 8ED0  0D67  2981         POPL    R3                       ; Restore reg
           05  0D6A  2982         RSB                              ; Done
                     0D6B  2983                                      ;
                     0D6B  2984 NET$MOV_CSTR::                        ; Move counted string with count byte
           35   BB  0D6B  2985         PUSHR   #^M<R0,R2,R4,R5>         ; Save regs
                     0D6D  2986                                      ;
     50    61   9B  0D6D  2987         MOVZBW  (R1),R0                  ; Get string length
           50   B6  0D70  2988         INCW    R0                       ; Include count itself
           05   11  0D72  2989         BRB     10$                      ; Continue in common
                     0D74  2990                                      ;
                     0D74  2991 NET$MOV_USTR::                        ; Mov counted str w/o count byte
           35   BB  0D74  2992         PUSHR   #^M<R0,R2,R4,R5>         ; Save regs
                     0D76  2993                                      ;
     50    81   9B  0D76  2994         MOVZBW  (R1)+,R0                 ; Get count value, advance ptr
63   61    50   28  0D79  2995 10$:    MOVC3   R0,(R1),(R3)             ; Move the string
                     0D7D  2996                                      ;
           35   BA  0D7D  2997         POPR    #^M<R0,R2,R4,R5>         ; Restore regs
           05   05  0D7F  2998         RSB
                     0D80  2999
                     0D80  3000                 .DSABL LSB
                     0D80  3001
```

```
                          0D80  3003 .SBTTL  NET$POST_IO     - Send IRP to COM$POST
                          0D80  3004 ;+
                          0D80  3005 ;
                          0D80  3006 ;  INPUTS:       R3      IRP address
                          0D80  3007 ;                R0      Scratch
                          0D80  3008 ;
                          0D80  3009 ;  OUTPUTS:      R0      SS$_NORMAL
                          0D80  3010 ;
                          0D80  3011 ;                All other registers are preserved
                          0D80  3012 ;-
                          0D80  3013 NET$POST_IO::                            ; Send IRP to COM$POST
                          0D80  3014                  ;
                          0D80  3015                  ;
                          0D80  3016                  ;  Complete the I/O
                          0D80  3017                  ;
                          0D80  3018                  ;
           55    DD       0D80  3019                  PUSHL   R5              ; Save XWB pointer
     55    1C A3    D0     0D82  3020                  MOVL    IRP$L_UCB(R3),R5    ; Get UCB address
00000000'GF      16       0D86  3021                  JSB     G^COM$POST          ; Another packet for the heap
        50    01    D0     0D8C  3022                  MOVL    S^#SS$_NORMAL,R0    ; Always return success
           55 8ED0    D0     0D8F  3023                  POPL    R5              ; Recover XWB address
                05    05     0D92  3024                  RSB                     ; Done
                          0D93  3025
                          0D93  3026
                          0D93  3027
                          0D93  3028
                          0D93  3029 .END
```

```
$$$                     = 00000020 R      02
$$OP                    = 00000002
$$_NSPMSG               = 00000000
$$_TR3MSG               = 00000000
$$_TR4MSG               = 00000000
ABORT                     000009AB R      03
ABORT_ALL                 00000A08 R      03
ACB$B_RMOD              = 0000000B
ACB$C_LENGTH            = 0000001C
ACB$L_KAST              = 00000018
ACB$L_PID               = 0000000C
ACP$ACCESSNET             ********    X   03
ACP$C_STA_F             = 00000004
ACP$C_STA_H             = 00000005
ACP$C_STA_I             = 00000000
ACP$C_STA_N             = 00000001
ACP$C_STA_R             = 00000002
ACP$C_STA_S             = 00000003
ACP$DEACCESS              ********    X   03
ACP$MODIFY                ********    X   03
ACT$ABORT                 ********    X   03
ACT$BUG                   0000047A R      03
ACT$CANLNK                ********    X   03
ACT$CONFIRM               00000618 RG     03
ACT$DEACCESS              0000078C RG     03
ACT$ENT_RUN               0000070B RG     03
ACT$INITIATE              0000064F RG     03
ACT$LOG                   0000047E R      03
ACT$NOLINK                00000484 R      03
ACT$NOP                   00000479 R      03
ACT$RCV_CA                ********    X   03
ACT$RCV_CC                ********    X   03
ACT$RCV_CI                ********    X   03
ACT$RCV_CR                ********    X   03
ACT$RCV_DATA              ********    X   03
ACT$RCV_DTACK             ********    X   03
ACT$RCV_DX                ********    X   03
ACT$RCV_LI                ********    X   03
ACT$RCV_LIACK             ********    X   03
ACT$RCV_RTS               ********    X   03
ACT$RES_DISC              000007B2 RG     03
ACT$RTS_NLT               ********    X   03
ACT$SHRLNK                0000048B R      03
ACT$SSABORT               0000048B R      03
ACT$_ABORT              = 0000000E
ACT$_BUG                = 00000000  G
ACT$_CANLNK             = 0000000F
ACT$_CONFIRM            = 00000013
ACT$_DEACCESS           = 00000015
ACT$_ENT_RUN            = 00000007
ACT$_INITIATE           = 00000011
ACT$_LOG                = 00000000
ACT$_NOP                = 00000004
ACT$_RCV_CA             = 00000003
ACT$_RCV_CC             = 00000005
ACT$_RCV_CI             = 00000002
ACT$_RCV_CR             = 00000001
ACT$_RCV_DATA           = 00000009
ACT$_RCV_DTACK          = 0000000A
ACT$_RCV_DX             = 0000000D
ACT$_RCV_LI             = 0000000B
ACT$_RCV_LIACK          = 0000000C
ACT$_RCV_RTS            = 00000008
ACT$_RES_DISC           = 00000010
ACT$_RTS_NLT            = 00000006
ACT$_SHRLNK             = 00000014
ACT$_SSABORT            = 00000012
ACT_DISPATCH              00000447 R      03
AQB$L_ACPPID            = 0000000C
AQB$L_ACPQBL            = 00000004
AT$_NULL                  ********    X   02
BIT...                  = 00000004
BRDCST                    00000A53 R      03
BUG$_NETNOSTATE           ********    X   03
CHANGE_STA                0000036F R      03
CHKRETADDR                00000345 R      03
CHK_X_IRP                 000007D1 R      03
CLEANUP_ACCESS            00000814 R      03
CNF$_ADVANCE            = 00000000
CNF$_QUIT               = 00000002
CNF$_TAKE_CURR          = 00000003
CNF$_TAKE_PREV          = 00000001
COM$POST                  ********    X   03
CRB$L_INTD              = 00000024
CRE_LNK                   000009DF R      03
CXB$B_R_AREA              00000039
CXB$B_R_FLG               00000038
CXB$B_R_NSPTYP            00000039
CXB$B_TYPE              = 0000000A
CXB$B_X_NSPTYP            0000004E
CXB$C_DYL               = 00000020
CXB$C_HEADER            = 00000048
CXB$C_R_LENGTH          = 0000003C
CXB$L_LINK              = 00000010
CXB$L_R_MSG               0000002C
CXB$L_R_RCB               00000028
CXB$T_DYL               = 00000028
CXB$T_X_DATA              00000057
CXB$T_X_XPORT             00000048
CXB$W_R_ADJ               0000003A
CXB$W_R_BCNT              00000030
CXB$W_R_DSTNOD            00000034
CXB$W_R_NSPSEQ            0000003A
CXB$W_R_PATH              00000032
CXB$W_R_SRCNOD            00000036
CXB$W_SIZE              = 00000008
CXB$W_X_NSPACK            00000053
CXB$W_X_NSPLOC            00000051
CXB$W_X_NSPREM            0000004F
CXB$W_X_NSPSEQ            00000055
DDB$L_DDT               = 0000000C
DDB$T_NAME              = 00000014
DEAL_ICB                  00000861 R      03
DECLARE                   00000AA5 R      03
```

```
DEV$M_AVL                ********    X   02
DEV$M_IDV                ********    X   02
DEV$M_MBX                ********    X   02
DEV$M_NET                ********    X   02
DEV$M_ODV                ********    X   02
DISC_ONE                 000009ED R      03
DLLTRN                   00000AD8 R      03
DPT$B_FLAGS              = 0000000D
DPT$C_LENGTH             = 00000038
DPT$C_VERSION            = 00000004
DPT$INITAB               00000038 R      02
DPT$M_NOUNLOAD           = 00000004
DPT$REINITAB             00000074 R      02
DPT$TAB                  00000000 R      02
DRAIN_RCV                00000921 R      03
DRAIN_XMT                000008C2 R      03
DYN$C_CRB                = 00000005
DYN$C_CXB                = 0000001B
DYN$C_DDB                = 00000006
DYN$C_DPT                = 0000001E
DYN$C_NDB                = 0000001C
DYN$C_ORB                = 00000049
DYN$C_UCB                = 00000010
EXE$ABORTIO              ********    X   03
EXE$ALONONPAGED          ********    X   03
EXE$DEANONPGDSIZ         ********    X   03
EXE$FINISHIO             ********    X   03
EXE$FORK                 ********    X   03
EXE$GQ_SYSTIME           ********    X   03
EXE$INSTIMQ              ********    X   03
EXE$WRTMAILBOX           ********    X   03
EXIT                     000009BA R      03
FKB$C_LENGTH             = 00000018
FUNCTABLE                00000038 R      03
FUNCTAB_LEN              = 00000058
GET_P1DSC                000007F2 R      03
GET_P2DSC                000007F7 R      03
GET_P3DSC                000007FC R      03
GET_P4DSC                00000801 R      03
GET_WNDSC                000007EE R      03
ICB$B_DATA               = 0000007C
ICB$B_RID                = 00000092
ICB$C_RID                = 00000010
ICB$T_RID                = 00000093
ICB$W_DLY_FACT           = 0000000E
ICB$W_DLY_WGHT           = 00000010
ICB$W_LOCLNK             = 00000002
ICB$W_PATH               = 00000000
ICB$W_RETRAN             = 0000000C
ICB$W_SEGSIZ             = 00000012
ICB$W_TIM_INACT          = 00000006
ICB$W_TIM_OCON           = 00000004
INIT_XWB                 00000C1F R      03
IO$M_FCODE               = 0000003F
IO$M_INTERRUPT           = 00000040
IO$V_ABORT               = 00000008
IO$_ACCESS               = 00000032
IO$_ACPCONTROL           = 00000038
IO$_DEACCESS             = 00000034
IO$_READLBLK             = 00000021
IO$_READVBLK             = 00000031
IO$_SETMODE              = 00000023
IO$_VIRTUAL              = 0000003F
IO$_WRITELBLK            = 00000020
IO$_WRITEVBLK            = 00000030
IOC$INITIATE               ********    X   03
IOC$MNTVER                 ********    X   03
IOC$REQCOM                 ********    X   03
IOC$RETURN                 ********    X   03
IPL$_SYNCH               = 00000008
IPL$_TIMER               = 00000008
IRP$C_BCNT               = 00000032
IRP$L_DIAGBUF            = 0000004C
IRP$L_IOQFL              = 00000000
IRP$L_IOST1              = 00000038
IRP$L_PID                = 0000000C
IRP$L_SVAPTE             = 0000002C
IRP$L_UCB                = 0000001C
IRP$L_WIND               = 00000018
IRP$M_FUNC               = 00000002
IRP$Q_NT_PRVMSK          = 00000040
IRP$V_COMPLX             = 00000003
IRP$W_BCNT               = 00000032
IRP$W_CHAN               = 00000028
IRP$W_FUNC               = 00000020
IRP$W_STS                = 0000002A
JIB$L_BYTCNT             = 00000020
JIB$L_BYTLM              = 00000024
JIB$W_FILCNT             = 00000030
LPD$B_PTH_INX            = 00000020
LPD$C_LOC_INX            = 00000001
LSB                      = 00000000
LSB$B_R_CXBCNT           = 00000028
LSB$B_R_CXBQUO           = 00000029
LSB$B_SPARE              = 0000002A
LSB$B_STS                = 0000002B
LSB$B_X_ADJ              = 0000000B
LSB$B_X_CXBACT           = 0000000D
LSB$B_X_CXBCNT           = 0000000F
LSB$B_X_CXBQUO           = 0000000E
LSB$B_X_PKTWND           = 0000000C
LSB$B_X_REQ              = 0000000A
LSB$L_CROSS              = 0000002C
LSB$L_R_CXB              = 00000020
LSB$L_R_IRP              = 0000001C
LSB$L_X_CXB              = 00000018
LSB$L_X_IRP              = 00000014
LSB$L_X_PND              = 00000010
LSB$M_BOM                = 00000020
LSB$M_EOM                = 00000040
LSB$M_LI                 = 00000001
LSB$S_LSB                = 00000030
LSB$S_SPARE              = 00000004
LSB$S_STS                = 00000001
```

```
LSB$V_BOM                       =  00000005
LSB$V_EOM                       =  00000006
LSB$V_LI                        =  00000000
LSB$V_SPARE                     =  00000001
LSB$W_HAA                       =  00000008
LSB$W_HAR                       =  00000006
LSB$W_HAX                       =  00000026
LSB$W_HNR                       =  00000024
LSB$W_HXS                       =  00000004
LSB$W_LNX                       =  00000002
LSB$W_LUX                       =  00000000
LTB$L_SLOTS                     =  00000010
LTB$L_XWB                       =  0000000C
LTB$W_SLT_TOT                   =  00000004
MASKH                           =  01000000
MASKL                           =  00000000
MBX$M_EVTAVL                    =  00000002
MBX$M_EVTRCVCHG                 =  00000004
MBX$M_EVTXMTCHG                 =  00000008
MBX$M_NETSTATE                  =  00000001
MBX$V_EVTAVL                    =  00000001
MBX$V_EVTRCVCHG                 =  00000002
MBX$V_EVTXMTCHG                 =  00000003
MBX$V_NETSTATE                  =  00000000
MBX_TABLE                          000001E8 R      03
MSG$_ABORT                      =  00000030
MSG$_CONNECT                    =  00000032
MSG$_DISCON                     =  00000033
MSG$_EVTAVL                     =  0000003E
MSG$_EVTRCVCHG                  =  0000003F
MSG$_EVTXMTCHG                  =  00000044
MSG$_EXIT                       =  00000034
MSG$_NETSHUT                    =  0000003B
MSG$_PATHLOST                   =  00000036
MSG$_REJECT                     =  00000038
MSG$_THIRDPARTY                 =  00000039
NET$AB_STTAB                       0000013C R      03
NET$ACCESS                         000005B1 R      03
NET$ACK_XMT_SEGS                   ******** X      03
NET$ACP_COMM                       00000969 RG     03
NET$ALONONPAGED                    00000D34 RG     03
NET$ALONPGD_Z                      00000D23 RG     03
NET$ALTENTRY                       ******** X      03
NET$AW_FLG_CLRM                    0000012C R      03
NET$AW_FLG_SETM                    0000011C R      03
NET$AZ_DR_CONTAB                   00000262 R      03
NET$AZ_DR_TABLE                    00000204 R      03
NET$CANCEL                         0000087B R      03
NET$CHK_X_IDLE                     000007C2 RG     03
NET$CMPL_ACC                       000006CA RG     03
NET$COMPLEX_EV                     00000330 RG     03
NET$CONTROL                        0000054A R      03
NET$CREATE_XWB                     00000BC6 RG     03
NET$CTLR_INIT                      000002E1 R      03
NET$C_ACTBITS                   =  00000005
NET$C_ACT_TIMER                 =  0000001E
NET$C_DR_ABORT                  =  00000009
NET$C_DR_ACCESS                 =  00000022
NET$C_DR_BUSY                   =  00000006
NET$C_DR_DEACC                  =  00000066  G
NET$C_DR_EXIT                   =  00000026
NET$C_DR_FMT                    =  00000005
NET$C_DR_INVALID                =  00000064  G
NET$C_DR_IVNODE                 =  00000002
NET$C_DR_NOBJ                   =  00000004
NET$C_DR_NONODE                 =  00000002
NET$C_DR_NOPATH                 =  00000027
NET$C_DR_NORMAL                 =  00000000
NET$C_DR_RSU                    =  00000001
NET$C_DR_SHUT                   =  00000003
NET$C_DR_THIRD                  =  00000008
NET$C_EFN_ASYN                  =  00000002
NET$C_EFN_WAIT                  =  00000001
NET$C_IPL                       =  00000008
NET$C_MAXACCFLD                 =  00000027
NET$C_MAXLINNAM                 =  0000000F
NET$C_MAXLNK                    =  000003FF
NET$C_MAXNODNAM                 =  00000006
NET$C_MAXOBJNAM                 =  0000000C
NET$C_MAX_AREAS                 =  0000003F
NET$C_MAX_LINES                 =  00000040
NET$C_MAX_NCB                   =  0000006E
NET$C_MAX_NODES                 =  000003FF
NET$C_MAX_OBJ                   =  000000FF
NET$C_MAX_WQE                   =  00000014
NET$C_MINBUFSIZ                 =  000000C0
NET$C_STABITS                   =  00000003
NET$C_TID_ACT                   =  00000003
NET$C_TID_RUS                   =  00000001
NET$C_TID_XRT                   =  00000002
NET$C_TRCTL_CEL                 =  00000002
NET$C_TRCTL_OVR                 =  00000005
NET$C_UTLBUFSIZ                 =  00001000
NET$DDT                            00000000 RG     03
NET$DEACCESS                       00000777 RG     03
NET$DEALLOCATE                     00000D4E RG     03
NET$DRAIN_FREE_CXB                 00000D16 RG     03
NET$END                            ******** X      02
NET$END_EVENT                      0000032A RG     03
NET$EVENT                          00000356 RG     03
NET$FDT_ACCESS                     000005A3 R      03
NET$FDT_CONTROL                    00000538 R      03
NET$FDT_DEACCESS                   00000711 RG     03
NET$FDT_RCV                        ******** X      03
NET$FDT_SETMODE                    0000050B R      03
NET$FDT_XMT                        ******** X      03
NET$FORK                           000002EC RG     03
NET$GL_OFF_DPTFLG                  00000118 RG     03
NET$GL_WORKBITS                    000001E4 RG     03
NET$GQ_PATCH                       00000090 RG     03
NET$INTERRUPT                      000002E1 R      03
NET$KAST                           ******** X      03
NET$MAP_R_REASON                   000002C0 RG     03
NET$MARK_LINK                      ******** X      03
```

```
                                                I  1
NET$MOV_CSTR                    00000D6B RG     03        NSP$$$_QUAL_FLW                 = 00000000
NET$MOV_TO_XWB                  00000D5F RG     03        NSP$$$_QUAL_INF                 = 00000000
NET$MOV_USTR                    00000D74 RG     03        NSP$$$_QUAL_MSG                 = 00000000
NET$M_MAXLNKMSK               = 000003FF                  NSP$$$_QUAL_SRV                 = 00000000
NET$M_STAMSK                  = 000000E0                  NSP$C_EXT_LNK                   = 0000001E
NET$POST_IO                     00000D80 RG     03        NSP$C_FLW_DATA                  = 00000000
NET$PRE_EMPT                    00000341 RG     03        NSP$C_FLW_INT                   = 00000001
NET$PURG_RUN                    000008AF RG     03        NSP$C_FLW_NOP                   = 00000000
NET$QUE_XWB                     00000CF1 RG     03        NSP$C_FLW_XOFF                  = 00000001
NET$RCV_DONE                    ********   X    03        NSP$C_FLW_XON                   = 00000002
NET$RESET_TIMER                 ********   X    03        NSP$C_HSZ_ACK                   = 00000007
NET$RET_SLOT                    00000CD1 RG     03        NSP$C_HSZ_CA                    = 00000003
NET$SCH_MSG                     000003A3 RG     03        NSP$C_HSZ_CC                    = 00000064
NET$SEND_CS_MBX                 00000B31 RG     03        NSP$C_HSZ_CD                    = 000000F0
NET$SEND_MBX                    00000B71 RG     03        NSP$C_HSZ_CI                    = 000000F0
NET$SETUP_RUN                   ********   X    03        NSP$C_HSZ_DATA                  = 00000009
NET$STARTIO                     00000490 R      03        NSP$C_HSZ_DC                    = 00000016
NET$TIMER                       ********   X    03        NSP$C_HSZ_DI                    = 00000016
NET$UNIT_INIT                   000002E2 R      03        NSP$C_HSZ_INT                   = 00000009
NET$UNSOL_INTR                  ********   X    03        NSP$C_HSZ_LS                    = 00000009
NET$XMT_DONE                    ********   X    03        NSP$C_INF_V31                   = 00000001
NET$XWB_LOCLNK                  00000CA9 RG     03        NSP$C_INF_V32                   = 00000000
NETEVT$_CA                    = 00000001   G               NSP$C_INF_V33                   = 00000002
NETEVT$_CANLNK                = 0000000D   G               NSP$C_MAXADR                    = 00000009
NETEVT$_CC                    = 00000002   G               NSP$C_MAX_DELAY                 = 00000014
NETEVT$_CCA                   = 00000010   G               NSP$C_MAX_R_CXB                 = 00000007
NETEVT$_CI                    = 00000000   G               NSP$C_MAX_XPW                   = 00000007
NETEVT$_CIA                   = 0000000F   G               NSP$C_MSG_CA                    = 00000024
NETEVT$_CRA                   = 00000011   G               NSP$C_MSG_CC                    = 00000028
NETEVT$_DATA                  = 00000005   G               NSP$C_MSG_CI                    = 00000018
NETEVT$_DC                    = 0000000B   G               NSP$C_MSG_DATA                  = 00000000
NETEVT$_DEA                   = 00000012   G               NSP$C_MSG_DC                    = 00000048
NETEVT$_DI                    = 0000000A   G               NSP$C_MSG_DI                    = 00000038
NETEVT$_DSCLNK                = 0000000C   G               NSP$C_MSG_DTACK                 = 00000004
NETEVT$_DTACK                 = 00000006   G               NSP$C_MSG_INT                   = 00000030
NETEVT$_INT                   = 00000008   G               NSP$C_MSG_LIACK                 = 00000014
NETEVT$_LIACK                 = 00000009   G               NSP$C_MSG_LS                    = 00000010
NETEVT$_LS                    = 00000007   G               NSP$C_SRV_MFC                   = 00000002
NETEVT$_MBXERR                = 00000013   G               NSP$C_SRV_NFC                   = 00000000
NETEVT$_PH2CCS                = 00000003   G               NSP$C_SRV_REQ                   = 00000001
NETEVT$_PROERR                = 00000014   G               NSP$C_SRV_SFC                   = 00000001
NETEVT$_RESDIS                = 0000000E   G               NSP$M_ACK_NAK                   = 00001000
NETEVT$_RTS                   = 00000004   G               NSP$M_ACK_NUM                   = 00000FFF
NETUPD$_ABOLNK                = 00000008                   NSP$M_ACK_VALID                 = 00008000
NETUPD$_ABORT                 = 00000001                   NSP$M_DATA_BOM                  = 00000020
NETUPD$_BRDCST                = 0000000A                   NSP$M_DATA_EOM                  = 00000040
NETUPD$_CONNECT               = 00000002                   NSP$M_DATA_OVFW                 = 00000080
NETUPD$_CRELNK                = 00000007                   NSP$M_FLW_CHAN                  = 0000000C
NETUPD$_DLL_ON                = 00000005                   NSP$M_FLW_DRV                   = 000000F0
NETUPD$_DSCLNK                = 00000009                   NSP$M_FLW_INT                   = 00000020
NETUPD$_EXIT                  = 00000003                   NSP$M_FLW_INUSE                 = 00000010
NETUPD$_PROCRE                = 00000004                   NSP$M_FLW_LISUB                 = 00000004
NETUPD$_REPLY                 = 0000000B                   NSP$M_FLW_MODE                  = 00000003
NEW_STATE                       0000037E R      03        NSP$M_FLW_SP1                   = 00000008
NSP$$$_QUAL_ACK               = 00000000                   NSP$M_FLW_SP2                   = 00000040
NSP$$$_QUAL_ALTFLW            = 00000000                   NSP$M_FLW_SP3                   = 00000080
NSP$$$_QUAL_DATA              = 00000000                   NSP$M_FLW_XOFF                  = 00000001
```

```
NSP$M_FLW_XON                   = 00000002
NSP$M_INF_VER                   = 00000003
NSP$M_MSG_INT                   = 00000020
NSP$M_MSG_LI                    = 00000010
NSP$M_SRV_01                    = 00000003
NSP$M_SRV_EXT                   = 00000080
NSP$M_SRV_FLW                   = 0000000C
NSP$M_SRV_REQ                   = 000000F3
NSP$M_SRV_SP1                   = 00000070
NSP$R_QUAL                      = 00000000
NSP$SOLICIT                     ******** X   03
NSP$S_ACK_NUM                   = 0000000C
NSP$S_ACK_SP2                   = 00000002
NSP$S_DATA_SP                   = 00000005
NSP$S_FLW_CHAN                  = 00000002
NSP$S_FLW_DRV                   = 00000004
NSP$S_FLW_MODE                  = 00000002
NSP$S_INF_VER                   = 00000002
NSP$S_MSG_SP1                   = 00000004
NSP$S_NSPMSG                    = 00000005
NSP$S_QUAL                      = 00000005
NSP$S_QUAL_ACK                  = 00000002
NSP$S_QUAL_ALTFLW               = 00000001
NSP$S_QUAL_DATA                 = 00000001
NSP$S_QUAL_FLW                  = 00000001
NSP$S_QUAL_INF                  = 00000001
NSP$S_QUAL_MSG                  = 00000005
NSP$S_QUAL_SRV                  = 00000001
NSP$S_SRV_01                    = 00000002
NSP$S_SRV_FLW                   = 00000002
NSP$S_SRV_SP1                   = 00000003
NSP$V_ACK_NAK                   = 0000000C
NSP$V_ACK_NUM                   = 00000000
NSP$V_ACK_SP2                   = 0000000D
NSP$V_ACK_VALID                 = 0000000F
NSP$V_DATA_BOM                  = 00000005
NSP$V_DATA_EOM                  = 00000006
NSP$V_DATA_OVFW                 = 00000007
NSP$V_DATA_SP                   = 00000000
NSP$V_FLW_CHAN                  = 00000002
NSP$V_FLW_DRV                   = 00000004
NSP$V_FLW_INT                   = 00000005
NSP$V_FLW_INUSE                 = 00000004
NSP$V_FLW_LISUB                 = 00000002
NSP$V_FLW_MODE                  = 00000000
NSP$V_FLW_SP1                   = 00000003
NSP$V_FLW_SP2                   = 00000006
NSP$V_FLW_SP3                   = 00000007
NSP$V_FLW_XOFF                  = 00000000
NSP$V_FLW_XON                   = 00000001
NSP$V_INF_VER                   = 00000000
NSP$V_MSG_INT                   = 00000005
NSP$V_MSG_LI                    = 00000004
NSP$V_MSG_SP1                   = 00000000
NSP$V_SRV_01                    = 00000000
NSP$V_SRV_EXT                   = 00000007
NSP$V_SRV_FLW                   = 00000002
NSP$V_SRV_SP1                   = 00000004
NSP$W_DSTLNK                    = 00000001
NSP$W_SRCLNK                    = 00000003
ORB$B_FLAGS                     = 0000000B
ORB$L_OWNER                     = 00000000
ORB$M_PROT_16                   = 00000001
ORB$W_PROT                      = 00000018
P1                              = 00000000
P2                              = 00000004
P3                              = 00000008
PATCH_AREA_SIZE                 = 00000080
PCB$L_JIB                       = 00000080
PCB$L_PHD                       = 0000006C
PCB$L_PID                       = 00000060
PHD$Q_PRIVMSK                   = 00000000
PR$_IPL                         = 00000012
PROCRE                            0000099E R     03
PROC_IO                           000004DA R     03
R3_OFF                          = 0000000C
R4_OFF                          = 00000010
R5_OFF                          = 00000014
RCB$B_ECL_DFA                   = 00000064
RCB$B_ECL_DWE                   = 00000065
RCB$B_ECL_RFA                   = 00000063
RCB$L_ACP_UCB                   = 00000014
RCB$L_AQB                       = 00000010
RCB$L_PTR_LTB                   = 00000024
RCB$L_PTR_TQE                   = 00000030
RCB$W_ADDR                      = 0000000E
RCB$W_CNT_MLL                   = 0000009E
RCB$W_CNT_XRE                   = 0000009C
RCB$W_ECLSEGSIZ                 = 0000007C
RCB$W_MAX_LNK                   = 00000058
RCB$W_MCOUNT                    = 00000054
RCB$W_TIM_CNI                   = 00000076
RCB$W_TRANS                     = 0000000C
REASON_C_LENGTH                 = 00000006   G
REASON_W_DR                     = 00000000   G
REASON_W_MBX                    = 00000004   G
REASON_W_SS                     = 00000002   G
REPLY                             00000AA0 R     03
SCH$GL_PCBVEC                     ********   X   03
SCH$WAKE                          ********   X   03
SETUP_XWB                         0000064F R     03
SIZ...                          = 00000001
SS$_ABORT                       = 0000002C
SS$_ACCVIO                      = 0000000C
SS$_BADPARAM                    = 00000014
SS$_CONNECFAIL                  = 000020DC
SS$_DEVALLOC                    = 00000840
SS$_FILNOTACC                   = 000000AC
SS$_ILLIOFUNC                   = 000000F4
SS$_INVLOGIN                    = 0000209C
SS$_LINKABORT                   = 000020E4
SS$_LINKDISCON                  = 000020EC
SS$_LINKEXIT                    = 000020F4
SS$_NOLINKS                     = 0000027C
```

```
SS$_NOMBX                       = 00000274
SS$_NORMAL                      = 00000001
SS$_NOSUCHNODE                  = 0000028C
SS$_NOSUCHOBJ                   = 000020A4
SS$_PATHLOST                    = 000020FC
SS$_PROTOCOL                    = 00002074
SS$_REJECT                      = 00000294
SS$_REMRSRC                     = 0000206C
SS$_SHUT                        = 0000208C
SS$_THIRDPARTY                  = 0000207C
SS$_UNREACHABLE                 = 00002094
TQE$B_RQTYPE                    = 0000000B
TQE$C_SSREPT                    = 00000005
TQE$L_FPC                       = 0000000C
TQE$L_FR4                       = 00000014
TQE$Q_DELTA                     = 00000020
TQE$V_REPEAT                    = 00000002
TR$C_MAXHDR                     = 0000001C
TR$C_NI_ALLEND1                 = 040000AB
TR$C_NI_ALLEND2                 = 00000000
TR$C_NI_ALLROU1                 = 030000AB
TR$C_NI_ALLROU2                 = 00000000
TR$C_NI_PREFIX                  = 000400AA
TR$C_NI_PROT                    = 00000360
TR$C_PRI_ECL                    = 0000001F
TR$C_PRI_RTHRU                  = 0000001F
TR$UPDATE                       ******** X   03
TR3$$$_QUAL_MSG                 = 00000000
TR3$$$_QUAL_RTFLG               = 00000000
TR3$C_MSZ_DATA                  = 00000006
TR3$C_MSG_DATA                  = 00000002
TR3$C_MSG_HELLO                 = 00000005
TR3$C_MSG_INIT                  = 00000001
TR3$C_MSG_NOP2                  = 00000008
TR3$C_MSG_ROUT                  = 00000007
TR3$C_MSG_STR2                  = 00000058
TR3$C_MSG_VERF                  = 00000003
TR3$M_MSG_CTL                   = 00000001
TR3$M_MSG_RTH                   = 00000002
TR3$M_RTFLG_PH2                 = 00000040
TR3$M_RTFLG_RQR                 = 00000008
TR3$M_RTFLG_RTS                 = 00000010
TR3$R_QUAL                      = 00000000
TR3$S_QUAL                      = 00000001
TR3$S_QUAL_MSG                  = 00000001
TR3$S_QUAL_RTFLG                = 00000001
TR3$S_RTFLG_012                 = 00000003
TR3$S_TR3MSG                    = 00000001
TR3$V_MSG_CTL                   = 00000000
TR3$V_MSG_RTH                   = 00000001
TR3$V_RTFLG_012                 = 00000000
TR3$V_RTFLG_5                   = 00000005
TR3$V_RTFLG_7                   = 00000007
TR3$V_RTFLG_PH2                 = 00000006
TR3$V_RTFLG_RQR                 = 00000003
TR3$V_RTFLG_RTS                 = 00000004
TR4$$$_QUAL_ADDR                = 00000000
TR4$$$_QUAL_RTFLG               = 00000000
TR4$$$_QUAL_SCLASS              = 00000000
TR4$C_BCE_MID1                  = 040000AB
TR4$C_BCE_MID2                  = 00000000
TR4$C_BCR_MID1                  = 030000AB
TR4$C_BCR_MID2                  = 00000000
TR4$C_BCT3MULT                  = 00000008
TR4$C_END_NODE                  = 00000003
TR4$C_HIORD                     = 000400AA
TR4$C_HSZ_DATA                  = 00000015
TR4$C_MSG_BCEHEL                = 0000000D
TR4$C_MSG_BCRHEL                = 0000000B
TR4$C_MSG_LDATA                 = 00000006
TR4$C_MSG_RDATA                 = 00000002
TR4$C_PRO_TYPE                  = 00000360
TR4$C_RTR_LVL1                  = 00000002
TR4$C_RTR_LVL2                  = 00000001
TR4$C_T3MULT                    = 00000002
TR4$C_VER_HIB                   = 00000000
TR4$C_VER_LOWW                  = 00000002
TR4$M_ADDR_AREA                 = 0000FC00
TR4$M_ADDR_DEST                 = 000003FF
TR4$M_RTFLG_INI                 = 00000020
TR4$M_RTFLG_LNG                 = 00000004
TR4$M_RTFLG_RQR                 = 00000008
TR4$M_RTFLG_RTS                 = 00000010
TR4$R_QUAL                      = 00000000
TR4$S_ADDR_AREA                 = 00000006
TR4$S_ADDR_DEST                 = 0000000A
TR4$S_QUAL                      = 00000002
TR4$S_QUAL_ADDR                 = 00000002
TR4$S_QUAL_RTFLG                = 00000001
TR4$S_QUAL_SCLASS               = 00000001
TR4$S_RTFLG_01                  = 00000002
TR4$S_RTFLG_VER                 = 00000002
TR4$S_SCLASS_57                 = 00000003
TR4$S_TR4MSG                    = 00000002
TR4$V_ADDR_AREA                 = 0000000A
TR4$V_ADDR_DEST                 = 00000000
TR4$V_RTFLG_01                  = 00000000
TR4$V_RTFLG_INI                 = 00000005
TR4$V_RTFLG_LNG                 = 00000002
TR4$V_RTFLG_RQR                 = 00000003
TR4$V_RTFLG_RTS                 = 00000004
TR4$V_RTFLG_VER                 = 00000006
TR4$V_SCLASS_1                  = 00000001
TR4$V_SCLASS_57                 = 00000005
TR4$V_SCLASS_BC                 = 00000004
TR4$V_SCLASS_LS                 = 00000002
TR4$V_SCLASS_METR               = 00000000
TR4$V_SCLASS_SUBA               = 00000003
UCB$B_DIPL                      = 0000005E
UCB$B_FIPL                      = 0000000B
UCB$C_LENGTH                    = 00000090
UCB$L_AMB                       = 00000060
UCB$L_DDB                       = 00000028
UCB$L_DEVCHAR                   = 00000038
```

```
UCB$L_DEVDEPEND         =  00000044
UCB$L_IOQFL             =  0000004C
UCB$L_IRP               =  00000058
UCB$L_LINK              =  00000030
UCB$L_VCB               =  00000034
UCB$M_BSY               =  00000100
UCB$M_ONLINE            =  00000010
UCB$M_TEMPLATE          =  00002000
UCB$V_BSY               =  00000008
UCB$W_DEVBUFSIZ         =  00000042
UCB$W_MB_SEED           =  00000000
UCB$W_STS               =  00000064
UCB$W_UNIT              =  00000054
UNKNOWN                    00000B2D R      03
VEC$L_ADP               =  00000014
VEC$L_INITIAL           =  0000000C
VEC$L_START             =  0000001C
VEC$L_UNITINIT          =  00000018
XWB                     =  00000000
XWB$$                   =  00000160  G
XWB$B_ACCESS            =  0000000B
XWB$B_ADJ_INX           =  00000124  G
XWB$B_DATA              =  0000005B
XWB$B_FIPL              =  0000001F
XWB$B_LOGIN             =  000000CC
XWB$B_LPRNAM            =  000000A4
XWB$B_PRO               =  0000005A
XWB$B_RID               =  0000006F
XWB$B_RPRNAM            =  000000B8
XWB$B_SP3               =  0000006E
XWB$B_STA               =  0000001E
XWB$B_TYPE              =  0000000A
XWB$B_X_FLW             =  0000006C
XWB$B_X_FLWCNT          =  0000006D
XWB$C_COMLNG            =  000000A4
XWB$C_CONLNG            =  00000112
XWB$C_DATA              =  00000010
XWB$C_LOGIN             =  00000040
XWB$C_LPRNAM            =  00000014
XWB$C_NDC_LNG           =  00000020
XWB$C_NUMSTA            =  00000008
XWB$C_RID               =  00000010
XWB$C_RPRNAM            =  00000014
XWB$C_STA_CAR           =  00000002
XWB$C_STA_CCS           =  00000004
XWB$C_STA_CIR           =  00000003
XWB$C_STA_CIS           =  00000001
XWB$C_STA_CLO           =  00000000
XWB$C_STA_DIR           =  00000006
XWB$C_STA_DIS           =  00000007
XWB$C_STA_RUN           =  00000005
XWB$L_DEA_IRP           =  00000104
XWB$L_FPC               =  00000020
XWB$L_FR3               =  00000024
XWB$L_FR4               =  00000028
XWB$L_ICB               =  0000010C
XWB$L_IRP_ACC           =  00000080
XWB$L_LINK              =  0000002C
XWB$L_ORGUCB            =  00000010
XWB$L_PID               =  00000034
XWB$L_PTR_RTHD          =  00000120  G
XWB$L_VCB               =  00000030
XWB$L_WLBL              =  00000004
XWB$L_WLFL              =  00000000
XWB$M_FLG_BREAK         =  00000001
XWB$M_FLG_CLO           =  00000200
XWB$M_FLG_IAVL          =  00001000
XWB$M_FLG_SCD           =  00000100
XWB$M_FLG_SDACK         =  00000008
XWB$M_FLG_SDFL          =  00004000
XWB$M_FLG_SDT           =  00000080
XWB$M_FLG_SIACK         =  00000004
XWB$M_FLG_SIFL          =  00002000
XWB$M_FLG_SLI           =  00000010
XWB$M_FLG_TBPR          =  00000800
XWB$M_FLG_WBP           =  00000040
XWB$M_FLG_WBUF          =  00000002
XWB$M_FLG_WDAT          =  00000400
XWB$M_FLG_WHGL          =  00000020
XWB$M_FLG_WMSK          =  0000039D
XWB$M_PRO_CCA           =  00000008
XWB$M_PRO_NAR           =  00000010
XWB$M_PRO_NFC           =  00000001
XWB$M_PRO_PH2           =  00000004
XWB$M_PRO_SFC           =  00000002
XWB$M_STS_ASTPND        =  00000400
XWB$M_STS_ASTREQ        =  00000800
XWB$M_STS_CON           =  00000010
XWB$M_STS_DIS           =  00000008
XWB$M_STS_DTNAK         =  00000100
XWB$M_STS_LINAK         =  00000200
XWB$M_STS_NDC           =  00001000
XWB$M_STS_OVF           =  00000080
XWB$M_STS_RBP           =  00000040
XWB$M_STS_SOL           =  00000004
XWB$M_STS_TID           =  00000001
XWB$M_STS_TLI           =  00000002
XWB$M_STS_TMO           =  00000020
XWB$Q_FORK              =  00000014
XWB$Q_FREE_CXB          =  00000118
XWB$R_CON_BLK           =  000000A4
XWB$R_RUN_BLK           =  000000A4
XWB$S_                  =  00000006
XWB$S_COMLNG            =  0000006E
XWB$S_CON_BLK           =  0000006E
XWB$S_DATA              =  00000010
XWB$S_DT                =  00000030
XWB$S_FLG               =  00000002
XWB$S_FORK              =  00000008
XWB$S_FREE_CXB          =  00000008
XWB$S_LI                =  00000030
XWB$S_LOGIN             =  0000003F
XWB$S_LPRNAM            =  00000013
XWB$S_NDC               =  00000020
```

```
XWB$S_PRO                       = 00000001
XWB$S_RID                       = 00000010
XWB$S_RPRNAM                    = 00000013
XWB$S_RUN_BLK                   = 00000064
XWB$S_STS                       = 00000002
XWB$S_XWB                       = 00000120
XWB$T_                          = 00000112
XWB$T_DATA                      = 0000005C
XWB$T_DT                        = 000000A4
XWB$T_LI                        = 000000D4
XWB$T_LOGIN                     = 000000CD
XWB$T_LPRNAM                    = 000000A5
XWB$T_RID                       = 00000070
XWB$T_RPRNAM                    = 000000B9
XWB$T_TR3HDR                    = 00000158   G
XWB$V_FLG_BREAK                 = 00000000
XWB$V_FLG_CLO                   = 00000009
XWB$V_FLG_IAVL                  = 0000000C
XWB$V_FLG_SCD                   = 00000008
XWB$V_FLG_SDACK                 = 00000003
XWB$V_FLG_SDFL                  = 0000000E
XWB$V_FLG_SDT                   = 00000007
XWB$V_FLG_SIACK                 = 00000002
XWB$V_FLG_SIFL                  = 0000000D
XWB$V_FLG_SLI                   = 00000004
XWB$V_FLG_TBPR                  = 0000000B
XWB$V_FLG_WBP                   = 00000006
XWB$V_FLG_WBUF                  = 00000001
XWB$V_FLG_WDAT                  = 0000000A
XWB$V_FLG_WHGL                  = 00000005
XWB$V_PRO_CCA                   = 00000003
XWB$V_PRO_NAR                   = 00000004
XWB$V_PRO_NFC                   = 00000000
XWB$V_PRO_PH2                   = 00000002
XWB$V_PRO_SFC                   = 00000001
XWB$V_STS_ASTPND                = 0000000A
XWB$V_STS_ASTREQ                = 0000000B
XWB$V_STS_CON                   = 00000004
XWB$V_STS_DIS                   = 00000003
XWB$V_STS_DTNAK                 = 00000008
XWB$V_STS_LINAK                 = 00000009
XWB$V_STS_NDC                   = 0000000C
XWB$V_STS_OVF                   = 00000007
XWB$V_STS_RBP                   = 00000006
XWB$V_STS_SOL                   = 00000002
XWB$V_STS_TID                   = 00000000
XWB$V_STS_TLI                   = 00000001
XWB$V_STS_TMO                   = 00000005
XWB$W_CI_PATH                   = 00000110
XWB$W_DELAY                     = 0000004E
XWB$W_DLY_FACT                  = 00000056
XWB$W_DLY_WGHT                  = 00000058
XWB$W_ELAPSE                    = 0000004A
XWB$W_FLG                       = 0000001C
XWB$W_LOCLNK                    = 0000003E
XWB$W_LOCSIZ                    = 00000040
XWB$W_PATH                      = 00000038
XWB$W_PROGRESS                  = 00000052
XWB$W_REFCNT                    = 0000000C
XWB$W_REMLNK                    = 0000003C
XWB$W_REMNOD                    = 0000003A
XWB$W_REMSIZ                    = 00000042
XWB$W_RETRAN                    = 00000054
XWB$W_R_REASON                  = 00000044
XWB$W_SIZE                      = 00000008
XWB$W_STS                       = 0000000E
XWB$W_TIMER                     = 00000050
XWB$W_TIM_ID                    = 00000048
XWB$W_TIM_INACT                 = 0000004C
XWB$W_X_REASON                  = 00000046
XWB$Z_NDC                       = 00000084
XWB_C_LEN                       = 0000017C
XWB_LOCLNK                        00000C97 R      03
_$ACT_DFLT                      = 00000000
_$ACT_INDEX                     = 00000016
_$EVENT_INDEX                   = 00000015
_$MSK                           = 00007DFF
_$TMP                           = 00000120   G
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000057 (87.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	0000008E (142.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00000D93 (3475.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	26	00:00:00.09	00:00:00.71
Command processing	157	00:00:01.15	00:00:04.92
Pass 1	990	00:00:45.45	00:01:28.11
Symbol table sort	0	00:00:05.56	00:00:12.19
Pass 2	520	00:00:10.73	00:00:19.59
Symbol table output	0	00:00:00.67	00:00:00.95
Psect synopsis output	2	00:00:00.05	00:00:00.15
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1698	00:01:03.71	00:02:06.71

The working set limit was 2000 pages.
243296 bytes (476 pages) of virtual memory were used to buffer the intermediate code.
There were 180 pages of symbol table space allocated to hold 3212 non-local and 320 local symbols.
3029 source lines were read in Pass 1, producing 35 object records in Pass 2.
91 pages of virtual memory were used to define 70 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	0
_255DUA28:[SHRLIB]EVCDEF.MLB;1	0
_255DUA28:[NETACP.OBJ]NETDRV.MLB;1	3
_255DUA28:[NETACP.OBJ]NET.MLB;1	10
_255DUA28:[SYS.OBJ]LIB.MLB;1	30
_255DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	53

3485 GETS were required to define 53 macros.

There were no errors, warnings or information messages.

```
MACRO/L:S=LIS$:NETDRVSES/OBJ=OBJ$:NETDRVSES MSRC$:NETDRVSES/UPDATE=(ENH$:NETDRVSES)+EXECML$/LIB+LIB$:NET/LIB+LIB$:NETDRV/LIB+SHRLIB$
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

NETDRVQRL
LIS

NETDRVSES
LIS

NETDRVNSP
LIS

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

NETDRVXPT
LIS

NETOPCOM
LIS

NETLLCNT
LIS

NETEVTLOG
LIS

NETPROCRE
LIS